UNIVERSITÉ DE MONTRÉAL

IMPLÉMENTATION D'UN PROCESSEUR VIDÉO SUR FPGA POUR LA DÉTECTION ET LA CORRECTION DE RÉFLEXIONS SPÉCULAIRES DANS DES IMAGES ENDOSCOPIQUES

RALPH-STÉPHANE TCHOULACK NGOUNOU

DÉPARTEMENT DE GÉNIE INFORMATIQUE ET GÉNIE LOGICIEL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE INFORMATIQUE)
AOUT 2008

Your file Votre référence
ISBN: 978-0-494-46083-2
Our file Notre référence
ISBN: 978-0-494-46083-2

UNIVERSITÉ DE MONTRÉAL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

IMPLÉMENTATION D'UN PROCESSEUR VIDÉO SUR FPGA POUR LA DÉTECTION ET LA CORRECTION DE RÉFLEXIONS SPÉCULAIRES DANS DES IMAGES ENDOSCOPIQUES

présenté par : TCHOULACK NGOUNOU Ralph-Stéphane
en vue de l'obtention du diplôme de : Maîtrise ès Sciences Appliquées
a été dûment accepté par le jury d'examen constitué de :

M.BILODEAU Guillaume-Alexandre, Ph.D., président
M.LANGLOIS J.M Pierre, Ph.D., membre et directeur de recherche
M.DAVID Jean-Pierre , Ph.D., membre

Remerciements

J'aimerais exprimer ma gratitude envers mon directeur de recherche Pierre Langlois, qui m'a guidé tout au long de cette maitrise. Sans son support, sa patience, son attention et ses connaissances, cette maitrise n'aurait pas pu arriver à terme.

Je remercie aussi ma famille qui m'a soutenu et encouragé durant les moments les plus pénibles. Elle m'a toujours poussé à donner le meilleur de moi, et je sais que sans cette famille extraordinaire, je n'aurais pas pu obtenir les mêmes résultats.

Je tiens aussi à remercier mes amis qui étaient toujours présents même quand je faisais passer les études avant eux. Ils m'ont soutenu et ont fait preuve de beaucoup de compréhension.

Résumé

Le cadre du projet se situe dans le milieu chirurgical. Lorsque le chirurgien désire opérer un patient, il peut le faire soit par une chirurgie invasive, soit par une chirurgie minimalement invasive. La chirurgie minimalement invasive se fait par endoscopie. Cela consiste à insérer les instruments chirurgicaux dans le corps du patient à l'aide d'incisions. Une petite caméra, l'endoscope, est aussi insérée afin de guider le chirurgien tout au long de l'opération.

La chirurgie par endoscopie a de nombreux avantages pour le patient. Elle entraîne moins de douleur, réduit la durée de l'hospitalisation et permet un prompt rétablissement. Cependant, elle rend la tâche difficile pour le chirurgien. Il est par exemple privé de la vision en trois dimensions puisque l'endoscope ne restitue qu'une vision en deux dimensions. Pour atténuer cette contrainte, des recherches ont été faites afin de créer un environnement de réalité augmentée, environnement dans lequel un modèle virtuel 3D de l'organe à opérer est superposé au model réel. Les étapes nécessaires à cette superposition requièrent une bonne segmentation des objets. Cependant la source lumineuse de l'endoscope, étant trop près des organes et des outils, réfléchit en provocant des artéfacts lumineux détruisant l'information utile et empêchant une bonne segmentation.

Le but de ce projet est de détecter les réflexions spéculaires présentes dans une séquence vidéo et les corriger. Ces deux étapes doivent être effectuées en temps réel afin de rendre

le traitement des séquences vidéos transparent au chirurgien. Afin d'atteindre les objectifs de temps réel, nous utilisons une architecture matérielle, Le FPGA.

Deux méthodes de détection sont utilisées pour localiser la position des réflexions spéculaires. La première utilise un histogramme de couleur afin de détecter l'intensité minimale des pixels spéculaires. La deuxième méthode se sert simultanément de deux plans (saturation et intensité) d'une trame. Elle consiste à effectuer un seuillage à la fois sur la trame de saturation et la trame en nuance de gris afin d'isoler la zone de réflexion spéculaire. La méthode de correction utilisée se sert du principe de restauration d'image. Il s'agit de prélever l'information provenant du contour de la zone à corriger et la propager à l'intérieur de cette zone. Afin de limiter l'utilisation des ressources du FPGA, plusieurs optimisations sont proposées pour l'implémentation de ces différentes méthodes.

Les objectifs de temps réel sont atteints. Le système implémenté détecte et corrige les réflexions avec un délai de 0.8 ms en utilisant 91% des LUTS disponibles

Abstract

The framework of the project is surgery. When operating a patient, the surgeon can either proceed by an invasive surgery or a minimally invasive surgery. The latter is made by endoscopy. It consists in inserting surgical instruments through cuts inside the patient's body. A small camera, the endoscope, is also inserted to guide the surgeon throughout the operation.

The endoscopic surgery has many advantages. In fact, it causes less pain, reduces the length of hospitalization, and allows for a quick recovery. However, it has some disadvantages for the surgeon. The endoscopic surgery enables solely a two-dimensional vision. Though, a three dimensions vision is more appropriate. To alleviate this constraint, research has been done to create an augmented reality surgical environment, an environment in which a virtual 3D model of the organ is superimposed on the real model. Therefore, this overlay requires an accurate segmentation of the objects. Note that, the light source of the endoscope, being too close to the organs and tools, reflects some luminous artefacts that damage useful information and prevent an accurate segmentation.

The aim of the project is to find out the specular reflections in a video sequence and to correct them. These two steps must occur in real time in order to transmit the video footage in a clear and a transparent way to the surgeon. The FPGA, a hardware architecture, is used to achieve these real time objectives.

Two detection methods are used to locate the position of the specular reflections. The first method consists in using a color histogram to detect the minimal intensity pixel of these reflections. The second method uses simultaneously two plans (saturation and intensity) of a frame. This method consists in making a threshold on both the saturation and intensity plans in order to isolate the specular area. The principle of image inpainting is used as the correction method. Indeed, it collects information around the area to correct, and spreads it inside that same area. To cope of with limited resources of the FPGA, several enhancements are proposed to implement these methods.

The real-time objectives are achieved. The system detects and corrects specularities within 0.8 ms and uses 91% of the available LUTs

Table des matières

Remerciements iv
Résumé v
Abstract vii
Table des matières ix
Liste des tableaux xi
Liste des figures xii
Sigles et abréviations xvi
Liste des annexes xvii
Introduction 1
Chapitre 1 Mise en contexte 4
1.1 Contexte chirurgical 4
1.1.1 Endoscopie 4
1.1.2 La réalité augmentée 6
1.2 Problèmes liés aux réflexions 7
1.3 Le FPGA par rapport au traitement d'images 9
1.4 Objectifs 12
Chapitre 2 Algorithmes pour la détection et la correction des réflexions spéculaires. 14
2.1 Algorithmes de détection 14
2.1.1 Approche unidimensionnelle 16
2.1.2 23
2.1.3 Approche bidimensionnelle 23
2.1.4 Détection du lobe spéculaire 25
2.2 Correction des réflexions spéculaires 29
2.2.1 Description générale 29
2.2.2 La restauration d'image ou image inpainting 30
2.3 Conclusion 34
Chapitre 3 Implémentation matérielle 35

3.1 Présentation générale du système 35
3.1.1 Description de l'environnement de prototypage 35
3.1.2 Signaux d'entrée du système de traitement 38
3.2 Contraintes liées à l'utilisation d'un FPGA 40
3.2.1 Parallélismes au sein d'un FPGA 40
3.2.2 Modes de traitements de données dans un FPGA 41
3.3 Architecture globale du système 43
3.4 Détection des réflexions spéculaires 46
3.4.1 Méthode unidimensionnelle 46
3.4.2 Méthode bidimensionnelle 58
3.4.3 Élargissement du masque 61
3.5 Correction linéaire 64
3.6 Lissage 68
3.7 Conclusion 70
Chapitre 4 Résultats et Discussion 71
4.1 Environnement de vérification 71
4.2 Vérification du système de détection 73
4.2.1 Détection monodimensionelle 73
4.2.2 Détection bidimensionnelle 79
4.3 Vérification du système de correction 82
4.4 Performances matérielles 85
4.5 Algorithme de traitement choisi 86
Conclusion 88
Références 91
Annexe 95

Liste des tableaux

Tableau 3-1 Ressources disponibles du FPGA XC2VP30 36

Tableau 3-2 Tableau des valeurs 49

Tableau 4-1 Valeurs maximale des plans S et M de l'image en figure 4.8 80

Tableau 4-2 Ressources utilisées pour chaque implémentation matérielle sur un FPGA Xilinx Virtex 2 pro XC2VP30 86

Liste des figures

Figure 1-1 Insertion d'instruments chirurgicaux et de l'endoscope(http://www.lasfce.com) 5

Figure 1-2 Étapes de la reconstitution dans le cadre de la réalité augmentée 7

Figure 1-3 Manifestations d'une réflexion diffuse (à gauche) et spéculaire (à droite) 8

Figure 1-4 Exemple de réflexions dans une image provenant d'une sonde endoscopique: on remarque bien la présence d'endroits très éblouissants (encerclés) 9

Figure 1-5 Différentes implémentations possibles pour un système de traitement d'images 12

Figure 2-1 Image dans le plan RGB (à gauche) et sa composante S correspondante (à droite) 17

Figure 2-2 Multiplication de l'image en figure 1 par la composante (1-S) (à gauche) et son plan Y (à droite); on constate que les réflexions se distinguent mieux dans l'image rehaussée 18

Figure 2-3 histogramme avant débruitage (en haut) et après débruitage (en bas) 20

Figure 2-4 Étapes de la détection du pic spéculaire 22

Figure 2-5 Masque spéculaire de l'image en figure 1 23

Figure 2-6 Plan M (à gauche) et plan S (à droite) de l'image en figure 1 24

Figure 2-7 Zone siège des réflexions spéculaires dans le diagramme MS selon [12] 25

Figure 2-8 Artéfact (en jaune) entourant une zone de pic spéculaire 26

Figure 2-9 Vue en 3D d'une réflexion. Observation du phénomène de montagne. En haut: Plan M à gauche, vue sur l'axe XY à droite. En bas: vue 3D. 27

Figure 2-10 Zone à corriger ([18]) 32

Figure 3-1 Environnement de prototypage 36

Figure 3-2 Planchette de développement XUV2P (www.digilentinc.com) 37

Figure 3-3 Evolution temporelle de Hsync et Vsync 39

Figure 3-4 Schéma global du système 44

Figure 3-5 Schéma bloc du système ; en vert : détection monodimensionnelle ; en turquoise : chemin détection bidimensionnelle 45

Figure 3-6 Histogrammes de l'image en figure 2.1 : histogrammes du plan Y sans rehaussement(en haut) et avec rehaussement (en bas). En bleu : histogramme non débruité ; en noir : histogramme débruité 48

Figure 3-7 Processus de mise à jour de la Ram contenant l'histogramme 51

Figure 3-8 Diagramme temporel présentant la décomposition en histogramme 52

Figure 3-9 Décomposition du signal par l'ondelette de Haar 54

Figure 3-10 Stockage des coefficients de la transformation en ondelettes 54

Figure 3-11 Recomposition du signal après débruitage 55

Figure 3-12 Noyau de convolution pour la dérivation 56

Figure 3-13 Deux étapes de dérivation 56

Figure 3-14 Diagramme MS de l'image en figure 2.1 60

Figure 3-15 Réflexions spéculaires détectées (en bleu) en appliquant les conditions 2.7 (à gauche) et 3.13 (à droite) 61

Figure 3-16 Résultat de l'élargissement du masque pour h=1 62

Figure 3-17 Principe de la fenêtre coulissante 63

Figure 3-18 Diagramme bloc de la fenêtre coulissante 63

Figure 3-19 Exemple d'élargissement du masque de h=1 pixel 64

Figure 3-20 Exemple d'interpolation linéaire 65

Figure 3-21 Schéma bloc de la correction linéaire d'une trame R,G ou B 67

Figure 3-22 Étapes de la correction linéaire 67

Figure 3-23 Diagramme temporel d'une interpolation linéaire 68

Figure 3-24 Correction par interpolation linéaire (en haut à droite), puis lissage (en bas) de l'image en haut à gauche 69

Figure 4-1 Flot de conception 71

Figure 4-2 Image aléatoire servant pour les tests 73

Figure 4-3 Résultats de la transformée en niveaux de gris sur matlab (à gauche) et sur modelsim (à droite) 73

Figure 4-4 Histogrammes résultant de la transformation en niveau de gris ; en haut: issu de matlab ; au milieu: issu de modelsim ; en bas: issu de la différence entre la transformation en niveau de gris et de modelsim et matlab 74

Figure 4-5 images utilisés pour tester la détection monodimensionnelle 76

Figure 4-6 Histogrammes obtenus à partir de Modelsim (en haut) et Matlab (en bas) pour l'image de droite de la figure 4.5 77

Figure 4-7 Histogrammes obtenus à partir de Modelsim (en haut) et Matlab (en bas) pour l'image de gauche de la figure 4.5 78

Figure 4-8 Histogrammes de l'image de gauche de la figure 4.5 obtenus à partir de Chiscope. Les débuts de bosse spéculaires sont calculés sur matlab 79

Figure 4-9 Image utilisée pour les tests du plan S et du plan M 80

Figure 4-10 Implémentation matlab(à gauche) et modelsim (à droite) du plan M de l'image en figure 4.9 81

Figure 4-11 Implémentation matlab(à gauche) et modelsim (à droite) du plan S l'image en figure 4.9 81

Figure 4-12 Implémentation matlab(à gauche) et modelsim (à droite) du masque spéculaire de l'image en figure 4.9 82

Figure 4-13 Manque d'efficacité du lissage pour de larges régions spéculaires; à gauche: image d'origine; à droite: image corrigée après implémentation matérielle. 83

Figure 4-14 Correction améliorée avec des étapes de lissage supplémentaire; à gauche: image d'origine; à droite: image corrigée avec 100 étapes de lissage 83

Figure 4-15 Correction de la réflexion spéculaire par la mauvaise couleur : la réflexion présente sur l'outil métallique est corrigé par l'information contenu dans la zone rouge à gauche de l'objet 84

Sigles et abréviations

FPGA	Field Programmable Gate Array
LUT	Lookup Table
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Scale Integrated Circuit
DSP	Digital Signal Processor
ASIC	Application Specific Integrated Circuit
RGB	plan de couleur rouge, vert, bleu (Red, Green, Blue)

Liste des annexes

Annexe 1: A video Stream Processor for Real-time Detection and Correction of Specular Reflections in Endoscopic Images

Introduction

Les avancées dans l'imagerie médicale ont joué un grand rôle dans le développement de chirurgies minimalement invasives dans une variété de procédures telles que la cardiologie, la neurochirurgie, l'orthopédie, l'urologie et l'oncologie. Cependant les difficultés inhérentes aux techniques minimalement invasives ont imposé des limites à leur applicabilité : contrôle d'instruments réduit, coordination mains-yeux inusuelle, vue réduite du champ d'opération, procurent des restrictions supplémentaires au chirurgien et requièrent une dextérité et une habileté considérables. D'un autre côté, ces procédures apportent beaucoup d'avantages au patient et au système de santé : incisions plus petites par rapport aux techniques d'opération usuelles, préservation du tissu, période de convalescence réduite.

L'utilisation de l'assistance vidéo a permis de faciliter l'opération et donc d'aider le chirurgien dans sa tâche. Elle consiste en une petite caméra appelée endoscope qui est insérée dans l'organisme à l'aide d'une petite fissure. L'endoscope est relié à un moniteur vidéo qui présente le site chirurgical. Cela permet au chirurgien de mieux orienter et de contrôler la position de ses instruments qui sont aussi insérés à l'aide de petites incisions. Pour une bonne marche de l'opération, la vidéo se doit d'être la plus nette possible. Cependant la source lumineuse, placée à proximité de la caméra, et donc très près des organes et des instruments, provoque un éblouissement lorsqu'elle réfléchit et conduit par la même occasion à une dégradation des conditions d'opération.

Augmenter la qualité vidéo semble donc nécessaire pour la bonne marche de l'opération, surtout dans la mesure où on aimerait constituer un environnement de réalité augmentée. Cela consisterait à détecter et atténuer les zones très éblouissantes à partir des informations provenant de l'endoscope, puis les retransmettre au moniteur vidéo. Ce processus devrait s'effectuer de manière discrète et rapide, de telle sorte que le chirurgien ne se rende pas compte qu'un traitement supplémentaire est effectué. En d'autres termes il devrait se faire en temps réel et utiliser le moins de mémoire possible pour augmenter sa portabilité afin d'être utilisable dans une salle d'opération et de s'intégrer au système d'opération. C'est sur cette base que nous avons décidé d'effectuer une implémentation matérielle d'un système de traitement d'image capable de détecter les zones de réflexion et les corriger. Nous étudions donc la faisabilité d'un tel système à l'aide d'un FPGA (Field Programmable Gate Array), un système numérique programmable permettant d'effectuer des conceptions matérielles de manière flexible.

La suite du mémoire sera organisée de la manière suivante :

Le chapitre 1 présentera la mise en contexte et les objectifs : mise en contexte sur l'endoscopie et la réalité augmentée et présentation du problème lié aux réflexions, mise en contexte sur les systèmes de traitement d'image et objectifs du projet

Le chapitre 2 traitera des algorithmes utilisés dans le cadre de la détection et de la correction des réflexions spéculaires; deux algorithmes de détection sont principalement décrits : un ayant une approche unidimensionnelle et l'autre ayant une approche bidimensionnelle; la correction quant à elle est faite à l'aide d'une méthode utilisant le principe de restauration d'image

Le chapitre 3 traitera de l'architecture du système à implémenter. À partir des différentes propositions énoncées dans le chapitre 2, des algorithmes qui en ressortent sont optimisés afin de permettre une implémentation matérielle efficace.

Le chapitre 4 présentera les résultats des algorithmes, et fera une comparaison des deux méthodes de détection implémentées.

Chapitre 1 Mise en contexte

1.1 Contexte chirurgical

1.1.1 Endoscopie

L'endoscopie peut être utilisée soit pour le diagnostic, soit pour traiter une maladie. Au départ (il y a un demi-siècle), elle était utilisée par les chirurgiens gynécologues à des fins diagnostiques pour explorer notamment des douleurs pelviennes [1].

Dans les années 80, quelques chirurgiens ont enlevé l'appendice (appendicectomie), la vésicule (cholécystectomie), puis progressivement pratiquement toutes les interventions de la coéliochirurgie (opération qui consiste à intervenir au niveau du pelvis ou de l'abdomen) ont été effectuées. D'autres spécialités l'ont ensuite adopté : la chirurgie thoracique (par thoracoscopie), l'urologie et plus récemment la chirurgie cardiovasculaire.

Dépendamment du type de chirurgie, un gaz inerte est inséré dans la paroi abdominale afin de créer un espace gazeux éloignant la paroi des viscères et facilitant la manipulation des instruments; ensuite, des instruments creux appelés trocarts sont mis en place à l'aide d'incisions de 5 à 10 mm; ces instruments permettent le passage de l'endoscope et d'autres instruments tels que la pince tractrice, le matériel de suture, l'aspiration et l'irrigateur. La figure 1.1 montre l'insertion de différents instruments et de l'endoscope dans l'organisme.

Figure 1-1 Insertion d'instruments chirurgicaux et de l'endoscope(http://www.lasfce.com)

L'endoscopie est qualifiée de minimalement invasive car elle a de multiples avantages par rapport à la chirurgie traditionnelle qui se fait par une grande incision (laparotomie) : elle donne moins de douleur, elle a moins de conséquences respiratoires, elle permet un rétablissement précoce ce qui réduit les complications du décubitus, elle réduit la durée d'hospitalisation permettant très souvent la chirurgie en ambulatoire, elle permet une récupération plus rapide [1].

Ce type de chirurgie nécessite cependant une formation adéquate, car elle entraîne de nouvelles difficultés pour le chirurgien :

- Il est privé de la vision en trois dimensions, puisque l'endoscope ne restitue qu'une vision en deux dimensions : il n'a donc pas la notion de relief, et est obligé de le déduire mentalement.
- Il est également privé de la possibilité de toucher les organes avec les mains : il n'a donc pas l'information tactile naturelle, mais seulement un retour de force qu'il perçoit à travers les instruments.

- Il doit introduire ses instruments seulement par quelques orifices, perdant ainsi la mobilité naturelle des instruments qu'il aurait en chirurgie classique.

1.1.2 La réalité augmentée

Pour réduire les complications liées à l'utilisation de l'endoscope, des recherches ont été faites afin de créer un environnement de réalité augmentée [2,3]. Il s'agit d'un environnement dans lequel un modèle virtuel 3D est superposé à la perception que nous avons naturellement, le tout en temps réel. En d'autres termes, il s'agit de la superposition d'images virtuelles aux images réelles, dans le cadre de la vision. Les principales étapes nécessaires à une superposition précise sont [4] :

- La création d'un modèle 3D de la structure anatomique interne de la partie où se déroulera l'intervention; cette structure est extraite grâce à des scans effectués avant l'opération (tomodensitométrie ou CT-scan); au cours de cette procédure, des marqueurs sont placés sur la peau du patient et sont visibles sur les scans.
- L'alignement du modèle 3D sur le patient : pour permettre un bon alignement, des marqueurs retro-réflectifs sont aussi placés sur le patient dans la salle d'opération. Des caméras infrarouges placées sur le plafond permettent de détecter ces marqueurs et de les faire correspondre à ceux placés sur le modèle 3D.
- Le suivi des instruments chirurgicaux : pour permettre la visualisation de la position des instruments par rapport au modèle 3D, les instruments doivent être détectés et suivi en temps réels.

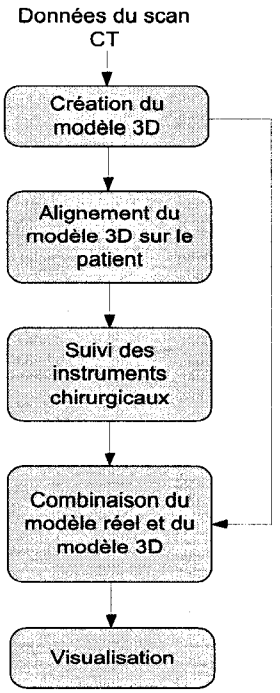


Figure 1-2 Étapes de la reconstitution dans le cadre de la réalité augmentée

1.2 Problèmes liés aux réflexions

Le chirurgien peut se retrouver facilement ébloui lorsqu'il pratique son opération à l'aide de l'endoscope car la source lumineuse peut se retrouver trop près des organes. De plus, la bonne marche des principales étapes résumées dans la figure 1.2 passe par une bonne segmentation des images endoscopiques pendant l'opération, donc en temps réel. Cependant la source lumineuse de l'endoscope, étant trop près des organes et des outils, est réfléchie en provoquant des artéfacts lumineux détruisant l'information utile et empêchant une segmentation adéquate. En effet, il sera facile pour un algorithme de segmentation de considérer une zone éblouissante comme étant une région à part entière alors qu'elle appartient à la même région que les pixels voisins.

La réflexion est le changement d'orientation d'une onde au contact d'une surface séparant deux milieux; elle peut être spéculaire ou diffuse (figure 1.3) en fonction du type de surface sur laquelle elle est réfléchie et de son angle d'incidence.

Une réflexion diffuse se produit lorsque la lumière se réfléchit dans une multitude de directions au contact d'une surface en général granuleuse ; dans ce cas, l'énergie incidente est répartie dans toutes les directions de réflexion. C'est ce type de réflexion qui permet de voir la couleur et la forme des objets.

Une réflexion spéculaire se produit lorsque l'onde incidente réfléchit dans une seule direction. Toute l'énergie incidente est donc réfléchie dans une seule direction, ce qui provoque un éblouissement lorsque la source lumineuse est proche de la surface de réflexion. Ce type de réflexion se produit au contact d'une surface lisse et/ou luisante.

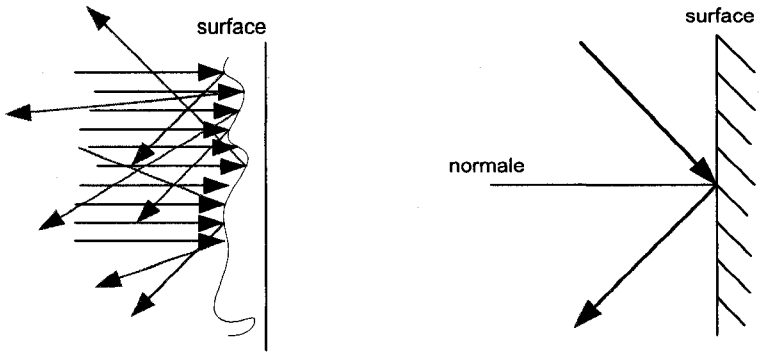


Figure 1-3 Manifestations d'une réflexion diffuse (à gauche) et spéculaire (à droite)

Il est donc utile de supprimer la composante spéculaire de la réflexion présente dans la vidéo provenant d'un système endoscopique afin d'améliorer son rendu. La figure 1.4 présente des endroits très éblouissants qui dégradent la qualité de l'image.

Figure 1-4 Exemple de réflexions dans une image provenant d'une sonde endoscopique: on remarque bien la présence d'endroits très éblouissants (encerclés)

1.3 Le FPGA par rapport au traitement d'images

Les systèmes de traitement de signaux peuvent être implémentés de deux manières différentes.

Le traitement peut se faire par microprocesseur. En général le concepteur d'un processeur construit un élément programmable utilisable pour une variété d'applications afin de maximiser la quantité d'éléments vendus. Étant donné que le concepteur ne sait pas quel programme sera implémenté, il crée une mémoire de programmes qui n'est pas fixe et un chemin de données assez général pour pouvoir supporter différents types

d'applications. Cela permet d'alléger grandement les contraintes d'implémentation. Utiliser un microprocesseur pour construire son programme se révèle donc assez facile car il n'est pas nécessaire de s'inquiéter de son implémentation matérielle. Il aura en plus une très grande flexibilité, car changer une fonctionnalité reviendrait juste à changer de programme et le recompiler; le prix unitaire sera aussi de faible coût pour de petites quantités. Cependant les performances pour des applications spécifiques, en l'occurrence le traitement de signal, ne seront pas optimales [5].

Pour augmenter le niveau de performance, il existe des microprocesseurs spécialement dédiés au traitement numérique de signal appelés DSP (Digital Signal Processors). Ils possèdent un chemin de données spécialisé, comportant des unités MAC (Multiply-Accumulate) accélérant les opérations de convolution et aussi du matériel spécial permettant de lire des données séquentiellement en mémoire tout en exécutant certaines opérations. Les DSPs procurent donc une certaine flexibilité tout en offrant une meilleure performance que les microprocesseurs généraux.

Mis à part l'implémentation par processeur général, il est possible d'utiliser des processeurs spécialisés pour le traitement de signaux. Les processeurs spécialisés sont des circuits numériques implémentés pour exécuter uniquement un programme spécifique. Ils ont donc une mémoire de programme fixe et un chemin de données spécialisé. Cela leur procure une très bonne performance, une faible taille mais le temps de design devient long et la flexibilité nulle. Ces processeurs peuvent être concus sur des ASICs (Application-Specific Integrated Circuits) : une fois le schéma électrique du circuit achevé, il est converti en éléments logiques puis envoyé en fonderie; la

conception sur ASICs entraîne cependant un coût de developpement très élevé. Ils peuvent aussi être concus sur des FPGAs (Field Programmable Gate Arrays).

Les FPGAs sont des matrices de blocs logiques connectés par un réseau d'interconnexion. Les blocs logiques et le réseau sont reprogrammables, ce qui permet d'implémenter de manière matérielle une application tout en préservant la capacité de changer différentes fonctionnalités de manière aisée. Les FPGAs offrent donc un compromis entre la flexibilité des microprocesseurs et la performance en vitesse des ASICs. Les gains en performance sont obtenus en éliminant l'étape « fetch-decode-execute » des microprocesseurs et en exploitant le parallélisme inhérent aux architectures matérielles. De plus, il est beaucoup plus facile d'effectuer des tests et simulations sur une architecture matérielle car on n'est pas en permanence confronté à des routines d'interruptions permettant d'accéder à des fonctions indépendantes du programme implémenté, ce qui est le cas sur les microprocesseurs. La figure 1.5 résume ces différentes architectures.

D'une manière générale, le traitement d'images en temps réel est difficile à atteindre sur un microprocesseur à cause de différents facteurs tels que la grande quantité de données que représente une image : pour une vidéo NTSC, les trames sont reçues à une fréquence de 30 trames par seconde, avec une résolution de 858x525 pixels. Pour exécuter des opérations sur chaque pixel en temps réel, le processeur doit effectuer 40.5 millions d'opérations à la seconde, sans tenir compte des étapes nécessaires au stockage et à la récupération des données provenant de l'extérieur. D'autre part, plusieurs applications

nécessitent que différents calculs soient effectués sur le même pixel, ce qui accroît encore plus le nombre d'opérations par seconde.

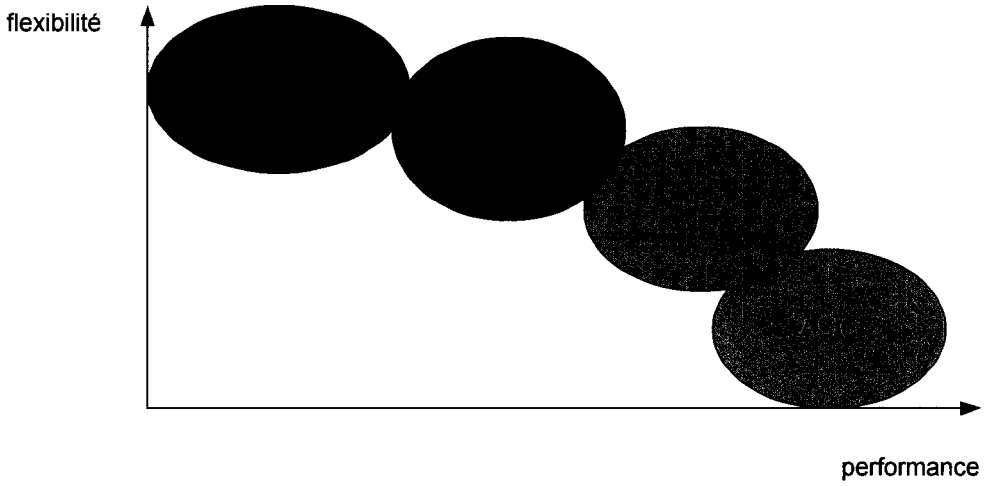


Figure 1-5 Différentes implémentations possibles pour un système de traitement d'images

Un bon compromis entre les différentes architectures présentées ci-dessus est donc le FPGA, malgré les contraintes liées à son utilisation : en général les programmes sont d'abord conçus de manière sérielle, sur un microprocesseur, puis transcrits pour une implémentation matérielle; la principale difficulté sera donc de transformer une application sérielle en une application matérielle en tenant compte des contraintes de bande passante, de temps, liés à la parallélisation des algorithmes.

1.4 Objectifs

L'objectif du projet est donc l'implémentation d'un système de traitement vidéo sur FPGA permettant la détection et la correction de réflexions spéculaires sur des images endoscopiques. Ce système doit être capable de fonctionner en temps réel afin d'aider le

chirurgien lors de l'opération d'un patient et d'être implémenté dans une salle à réalité augmentée. Plus en détails, nous devons donc :

- Trouver des algorithmes de détection et correction de réflexions spéculaires dans les images endoscopiques.
- Rendre ces algorithmes implémentables sur un FPGA à faible capacité, c'est-à-dire minimiser le coût de ces algorithmes en termes de ressources matérielles.
- Implémenter ces algorithmes en essayant d'atteindre le temps réel tout en conservant d'excellentes qualités visuelles.

Chapitre 2 Algorithmes pour la détection et la correction des réflexions spéculaires

Dans ce chapitre nous présentons deux types d'algorithmes de traitement, les algorithmes de détection et les algorithmes de correction. Après une revue de littérature des algorithmes existants, nous nous concentrons sur ceux que nous avons choisis pour effectuer notre implémentation. Nous présentons donc deux approches dans le cadre de la détection et différentes méthodes de correction que nous combinerons afin d'obtenir ceux que nous implémenterons.

2.1 Algorithmes de détection

Selon le modèle de réflexion dichromatique [6], une image est une combinaison linéaire de sa composante diffuse C_b et de sa composante spéculaire C_i:

$$C(x,y) = m_i C_i(x,y) + m_b C_b(x,y) \qquad (2.1)$$

où m_i et m_b sont des facteurs dépendant de la géométrie de la scène, de l'angle d'incidence de la source lumineuse, et de l'angle de vision de l'image. Il est donc possible de dissocier les deux composantes, et par la même occasion de ne retenir que la composante diffuse. Plusieurs techniques ont donc été proposées afin de retirer la composante spéculaire de l'image.

Certaines de ces méthodes se servent de plusieurs angles de vision de la même image. Par exemple, Lee et Bajcsy se servent d'une technique appelée « spectral differencing » [7]. Celle-ci se base sur le fait que l'irradiance (flux de radiation arrivant

sur une surface par unité d'aire) de l'image provenant d'une réflexion lambertienne (réflexion diffuse) ne change pas en fonction de l'angle de vision, contrairement à l'irradiance provenant d'une réflexion spéculaire. Il suffit donc, après analyse, de détecter les points inconsistants au travers des différents points de visions de l'image. Cette méthode n'est pas applicable dans le cas de l'endoscope car la source lumineuse est solidaire de la lentille. En plus de cela, elle serait difficile à utiliser en temps réel à cause de la complexité de ses calculs et la quantité de données à stocker pour une seule image.

Certaines méthodes adoptent une approche statistique, mais ne prennent pas en compte le processus de formation de l'image [8]. D'autres méthodes ayant plutôt une approche physique prennent bien en compte le processus de formation. C'est le cas de Schlüns et Koschan qui utilisent le cube RGB afin de déterminer le nombre de matériaux dans la scène, les segmentent, et ensuite les séparent respectivement entre leurs composantes diffuse et spéculaire [9]. Bien qu'efficace, cette approche ne fonctionne correctement que pour des images ayant peu de textures et un arrière-plan uniforme; en plus elle est coûteuse à cause du traitement en 3D de l'image. Elle serait donc difficile à implémenter pour une application temps réel.

Par contre, St-Pierre part du principe que le plan RGB est le mieux adapté pour détecter des réflexions et construit trois histogrammes pour chacune des composantes rouge, vert et bleu [10]. Il détermine ensuite la position commune aux trois histogrammes où il y a réflexion spéculaire.

En se basant sur les précédentes méthodes, nous étudions deux approches qui permettent de détecter les pixels spéculaires de manière dynamique (en se servant des informations provenant de l'image) : une approche à histogramme unidimensionnel et une approche bidimensionnelle.

2.1.1 Approche unidimensionnelle

2.1.1.1 Décomposition en histogrammes

Les histogrammes d'une trame de couleur représentent le nombre de pixels en fonction des intensités de cette couleur. Ils sont construits en divisant l'espace RGB en un certain nombre de cases ou plages d'intensité, puis en comptant le nombre de pixels de l'image dans chaque case. Ils permettent donc de diviser la trame en différentes régions d'intensité. On peut décomposer une trame en trois histogrammes, un pour chacune des couleurs du plan RGB. En plus, nous travaillons avec des séquences provenant de l'intérieur du corps humain, dominé par la couleur rouge. On peut donc supposer que la détection d'une couleur blanche revient à détecter une réflexion spéculaire. La couleur blanche étant formée des couleurs rouge, verte et bleue, la détection simultanée de ces trois couleurs dans une plage d'intensités élevées des histogrammes indique que tous les pixels faisant partie de cette région d'intensités sont des réflexions spéculaires [10].

Par ailleurs, on constate que la composante S du plan HSV permet de faire ressortir les réflexions spéculaires (figure 2.1).

Cette composante se calcule assez aisément à partir des composantes R, G, B :

$$S = \frac{\max(R,G,B) - \min(R,G,B)}{\max(R,G,B)} \tag{2.2}$$

Bien que la réflexion soit bien visible à l'œil nu, l'histogramme de cette composante ne permet pas de délimiter la région des réflexions spéculaires. Il est quand même possible d'utiliser la composante S afin de rehausser l'image dans le plan RGB et d'y mettre plus en évidence les réflexions spéculaires. Cela se fait en multipliant les canaux RGB par (1-S). La figure 2.2 permet d'observer les effets du rehaussement.

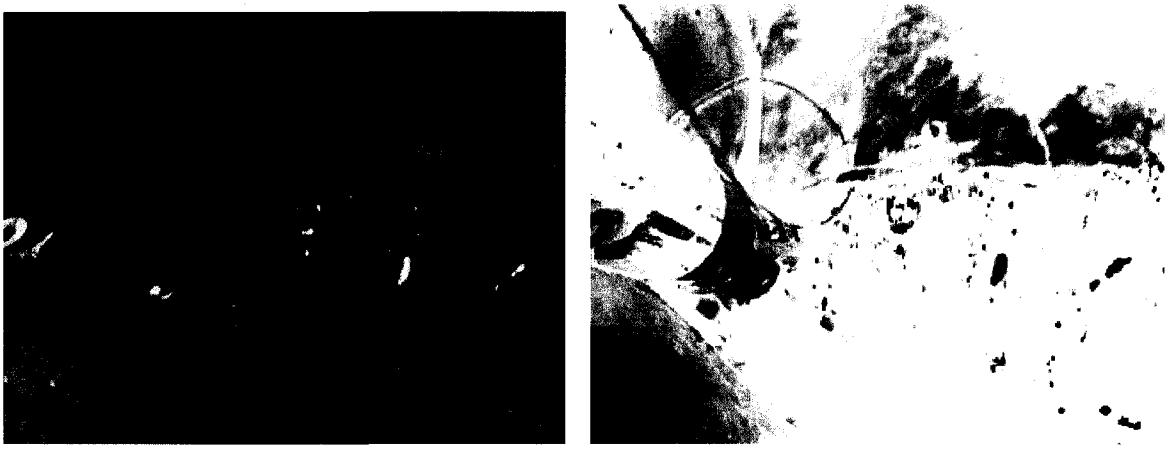


Figure 2-1 Image dans le plan RGB (à gauche) et sa composante S correspondante (à droite)

Le rehaussement permet de mieux délimiter les régions de réflexions spéculaires dans les histogrammes car il accentue la différence de valeur entre les zones d'intensité moyenne par rapport à la trame (en d'autre terme les zones de réflexion diffuse) et les zones de forte intensité. Cela entraine une bonne démarcation dans l'histogramme, les pixels provenant de la réflexion diffuse étant tous regroupés vers la gauche.

Gröger et al. observent que pour une image en niveaux de gris, une détection de réflexions spéculaires est largement possible par un simple seuillage car celles-ci ont des

intensités indépendantes des autres pixels de l'image [11]. L'image en niveaux de gris est calculée en combinant les composantes RGB. Il s'agit en fait de la composante Y du plan YUV, donnée par l'équation 2.3:

$$Y = 0.299R + 0.587G + 0.114B \tag{2.3}$$

La figure 2.2 présente l'image rehaussée transformée en nuance de gris.

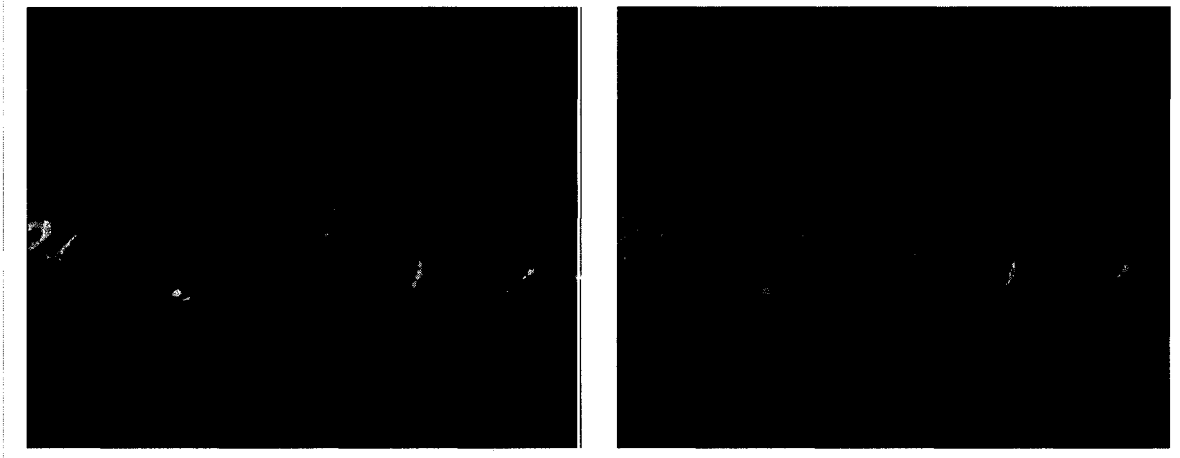


Figure 2-2 Multiplication de l'image en figure 1 par la composante (1-S) (à gauche) et son plan Y (à droite); on constate que les réflexions se distinguent mieux dans l'image rehaussée

2.1.1.2 Débruitage et détection du pic spéculaire

Comme décrit plus haut, l'histogramme est normalement divisé en deux grandes régions : une région où sont localisés la plupart des pixels, et une autre délimitant le pic spéculaire. Détecter la région spéculaire reviendrait donc à déterminer le début de la seconde région. Ce processus n'est pas aisé à cause de la nature bruitée de l'histogramme. En effet, la présence de plusieurs autres régions empêcherait l'algorithme de détection de délimiter la véritable région spéculaire. On peut constater en observant la figure 2.3 que notre algorithme de détection prendra le bruit comme

étant la véritable région spéculaire si un débruitage n'est pas effectué sur l'histogramme. Il convient donc de débruiter l'histogramme avant d'effectuer la détection. Pour effectuer le débruitage, St-Pierre constate que l'approche par transformation en ondelettes donne de meilleurs résultats qu'une approche fréquentielle ou une approche par approximation de fonctions continues [10].

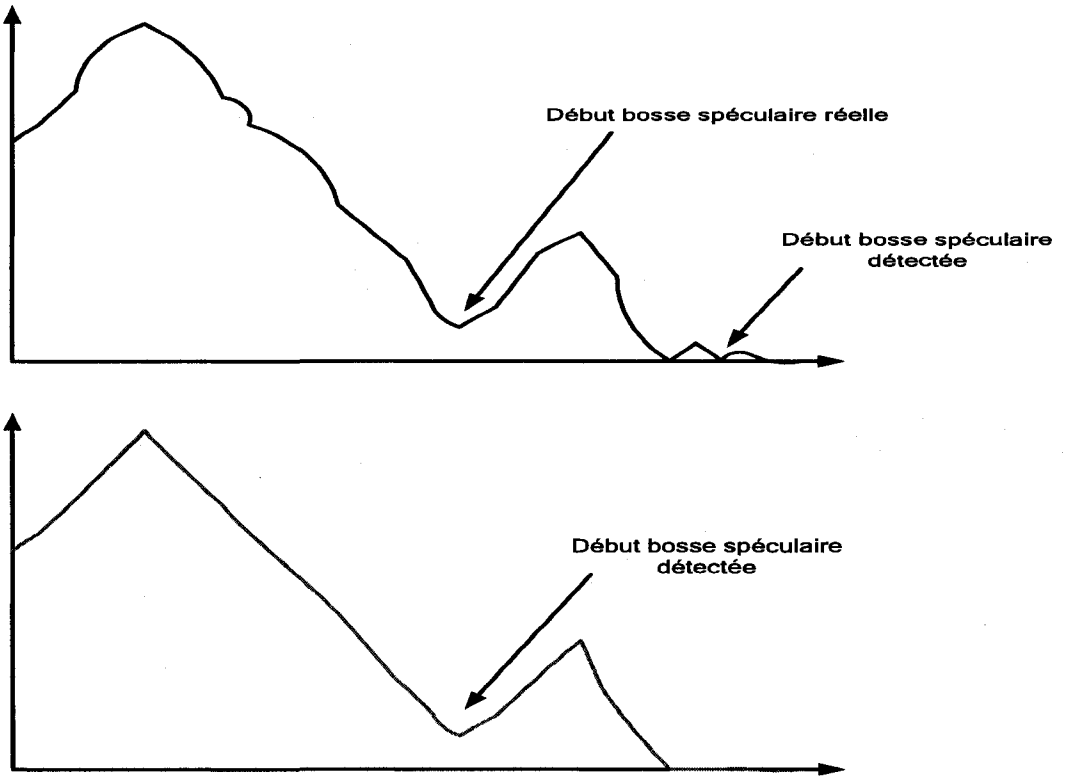


Figure 2-3 histogramme avant débruitage (en haut) et après débruitage (en bas)

La transformation en ondelettes permet de représenter le signal avec plus ou moins de détails. Elle transforme le signal en deux groupes de coefficients : les coefficients d'approximation conservant la forme même du signal, et les coefficients de détails, permettant de donner plus de détails au signal. Le bruit étant généralement stocké dans

les coefficients de détail, faire un débruitage du signal revient ainsi à effectuer une transformation en ondelettes, puis effectuer un seuillage des coefficients de détail. Ensuite il suffit d'effectuer la transformée inverse.

Il faudrait donc être capable de déterminer un seuil en dessous du quel on ne considère pas les petits coefficients ; Sudha et Al. présentent plusieurs méthodes de seuillages dont les principales sont le seuillage doux et le seuillage dur [22]. Le seuillage dur veut que les coefficients soient retenus uniquement s'ils dépassent une certaine valeur seuil; le seuillage doux réduit les coefficients au lieu de les mettre directement à 0 s'ils sont inférieurs à un certain seuil. La technique de seuillage que nous utilisons est celle du VisuShrink de Donoho et Johnstone [23]. C'est un seuillage dur qui utilise le seuil universel défini par l'équation 2.4 :

$$T = \sigma\sqrt{2\log_e M} \tag{2.4}$$

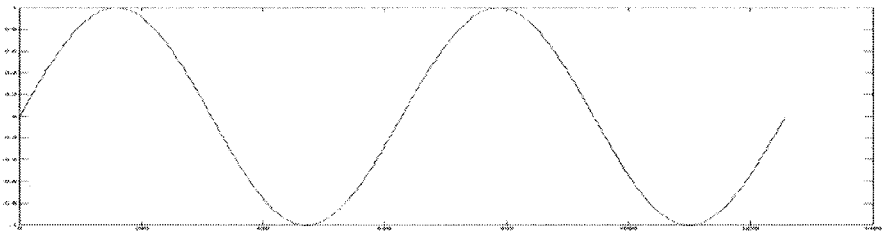
Où σ est la variance des coefficients de détail et M le nombre de données.

La détection du pic spéculaire consiste à déterminer le début de la dernière région de l'histogramme ; tous les pixels ayant des intensités supérieures à cette valeur feront partie du pic spéculaire. La détection se fait en deux étapes.

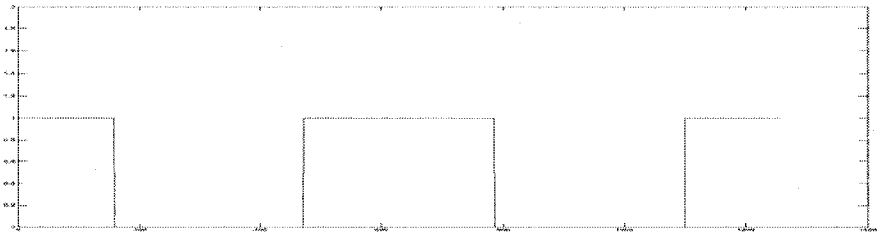
L'histogramme est dérivé une première fois. Si la valeur de la dérivée est supérieure à 0 on lui attribue la valeur maximale (255 si les données sont représentées sur 8 bits), sinon on lui attribue la valeur 0.

Il est ensuite dérivé une seconde fois, et le même procédé de seuillage est appliqué. Le résultat est une suite de pics dont le dernier correspond au début de la région spéculaire.

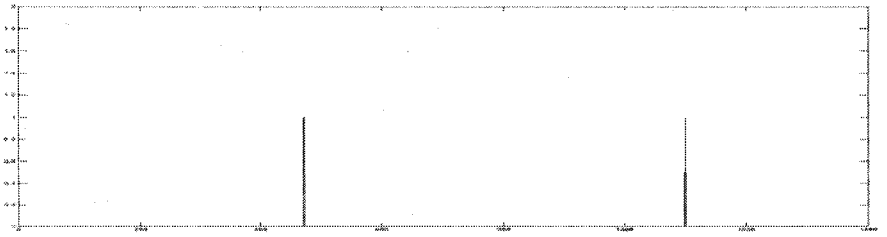
La figure 2.4, prise à titre d'exemple, représente les deux étapes de détection effectuées sur un signal sinusoïdal.



Signal débruité



Signal après la premiere dérivation



Signal après la deuxième dérivation

Figure 2-4 Étapes de la détection du pic spéculaire

Pour déterminer les zones de réflexion spéculaire dans l'image, il suffit donc de chercher tous les pixels de l'image dont l'intensité est supérieure ou égale à la valeur du début de la région spéculaire déterminée lors de la seconde dérivation. Les pixels faisant partie de la région spéculaire seront donc mis en blanc, et les autres en noir. À la fin on obtient un

masque spéculaire. Ce masque spéculaire indique la position exacte des régions spéculaires dans l'image (figure 2.5).

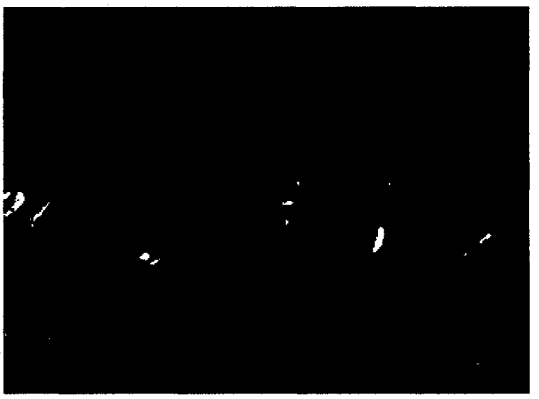


Figure 2-5 Masque spéculaire de l'image en figure 1

2.1.3 Approche bidimensionnelle

Il est possible de détecter la présence des réflexions spéculaires en adoptant une méthode bidimensionnelle. Cette méthode se base sur le fait que les réflexions spéculaires sont facilement visibles dans le plan S, ainsi que dans les images en nuances de gris. Il est donc possible de mettre en relation ces deux plans afin d'effectuer une bonne segmentation de l'image.

Ortiz et Torres construisent un diagramme appelé diagramme MS et recherchent les réflexions spéculaires dans une zone précise de ce diagramme [12]. Ce diagramme a pour abscisse M et pour ordonnée S ; la première étape consiste à déterminer le plan M (figure 2.6). La formule est la même que celle employée dans [12]. Elle correspond tout simplement à la moyenne des composantes RGB (équation 2.5):

$$M=\frac{1}{3}(R+G+B) \tag{2.5}$$

À partir de M, on détermine S (figure 2.6). Contrairement au plan S défini lors de l'approche précédente, celui-ci est normalisé par la norme L1 définie dans [13]. En effet Angulo et Serra constatent que cela permet de mieux délimiter les différentes régions de l'image (équation 2.6) :

$$S=\begin{cases}\frac{1}{2}(2R-G-B)=\frac{3}{2}(R-M), si(B+R)\geq 2G\\ \frac{1}{2}(R+G-2B)=\frac{3}{2}(M-B), si(B+R)<2G\end{cases} \tag{2.6}$$

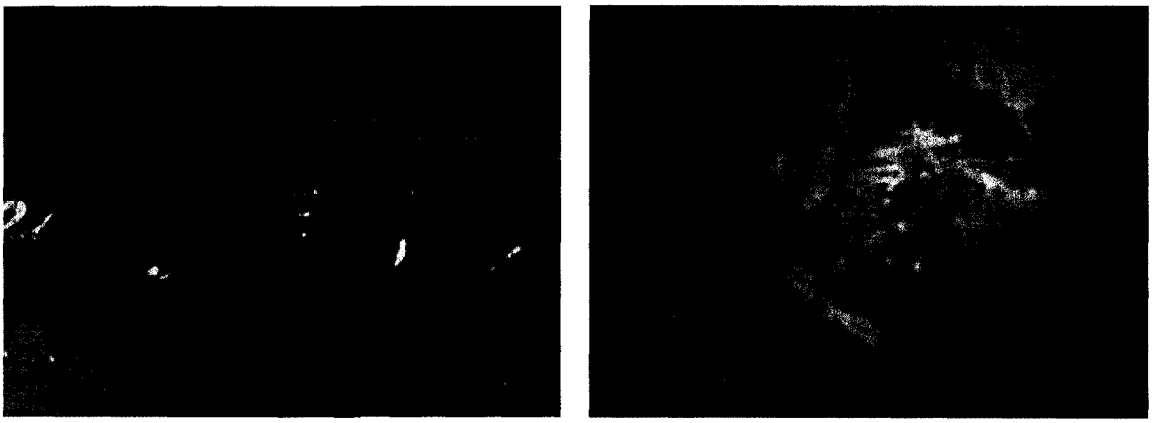


Figure 2-6 Plan M (à gauche) et plan S (à droite) de l'image en figure 1

Une fois le diagramme MS défini, Ortiz et Torres délimitent une zone précise et considèrent tous les pixels appartenant à cette zone comme étant des réflexions spéculaires ; étant donné qu'une réflexion spéculaire est naturellement caractérisée par une très grande intensité et une saturation très basse, cette zone située dans la partie

inférieure droite (figure 2.7) du diagramme est délimitée par les équations c_3, c_4 et s_1. La constante s_1 correspond à la zone de saturation des couleurs primaires G et B, c_3 et c_4 sont définis par l'équation 2.7

$$\begin{cases} c_3 = -\frac{3}{2}(m-255), \forall m \in [m_3, m_{\max}] \\ c_4 = -3(m-255), \forall m \in [m_4, m_{\max}] \end{cases} \tag{2.7}$$

avec $m_3 = \frac{2}{3} m_{\max}$, $m_4 = \frac{5}{6} m_{\max}$ et $m_{\max}$ la valeur maximale dans le plan M.

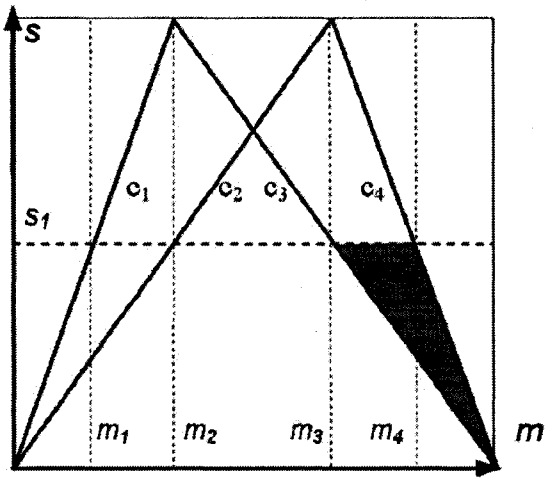


Figure 2-7 Zone siège des réflexions spéculaires dans le diagramme MS selon [12]

2.1.4 Détection du lobe spéculaire

La bosse spéculaire présente uniquement les zones de forte intensité, à savoir les pics spéculaires. Autour de ces pics, il existe des zones de transition allant jusqu'aux zones de réflexion diffuse, les lobes spéculaires. Par endroit on peut aussi détecter autour de

ces pics des artéfacts, de couleur noire ou jaune foncé. La figure 2.8 permet de mettre en évidence des artefacts de couleur jaune entourant une région spéculaire que nous avons colorée en couleur turquoise. Ces artéfacts sont souvent causés par la source lumineuse située trop près de la surface et par une saturation de la caméra.

Ces artéfacts ayant des couleurs qui ne peuvent être considérés ni comme diffuses, ni comme spéculaires devraient être retirés afin de faciliter la correction. En résumé il faudrait trouver un moyen d'inclure dans le masque spéculaire le lobe spéculaire et les artéfacts ; nous considérons dans la suite que les artéfacts font partie du lobe.

Bhattacharyya [14], puis Saint-Pierre [10] ont proposé dans leurs mémoires respectifs une approche commune dite descente en intensité.

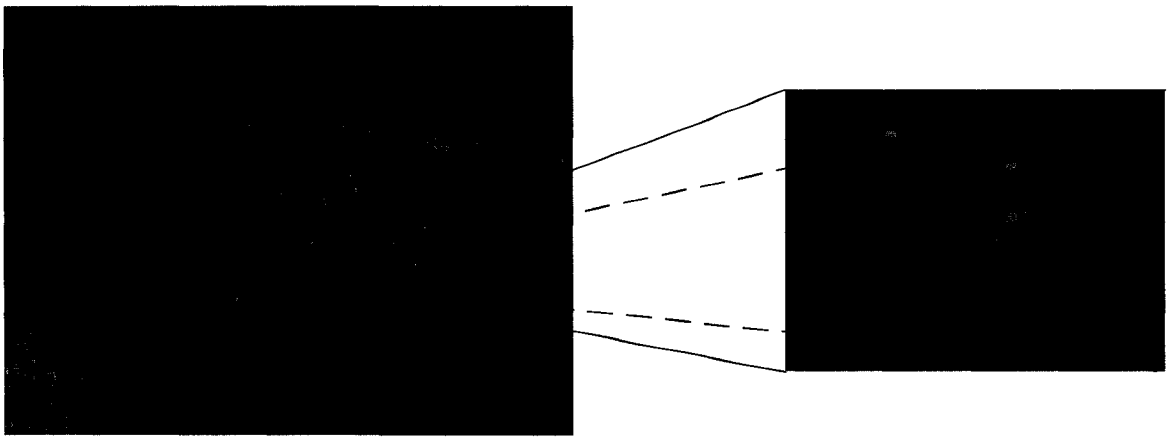


Figure 2-8 Artéfact (en jaune) entourant une zone de pic spéculaire

Pour détecter le lobe, ils partent du pic spéculaire et font une descente dans les dimensions horizontale et verticale jusqu'à ce qu'ils détectent la zone de réflexion diffuse. Les réflexions spéculaires, étant par définition plus brillantes que leur voisinage,

peuvent être représentées comme des montagnes avec pour sommet le pic spéculaire et pour base la zone de réflexion diffuse. Dans la figure 2.9, une représentation en 3D d'une réflexion prise sur le plan M de la figure 2.1 illustre ce phénomène ; la base est caractérisée par une variation d'intensité très légère, contrairement à la « montagne » proprement dite dont la variation est beaucoup plus abrupte.

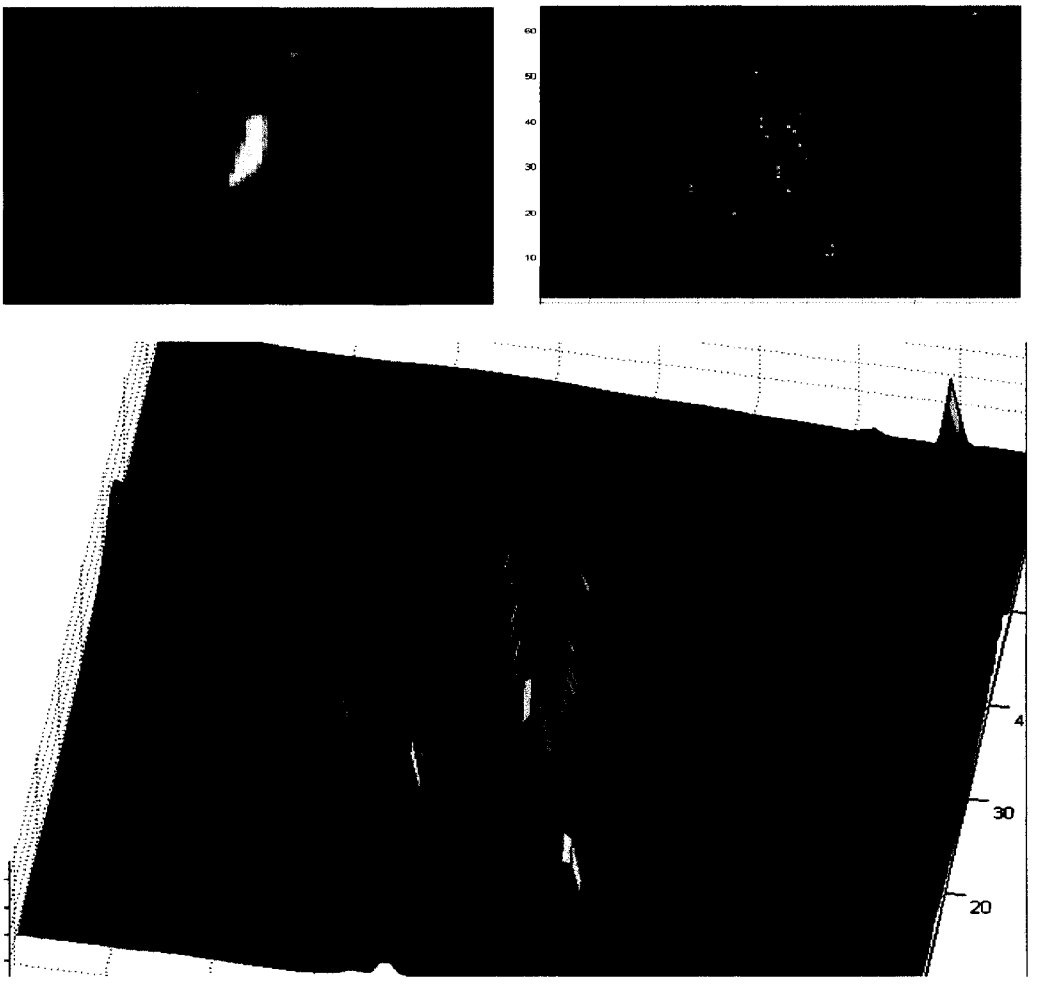


Figure 2-9 Vue en 3D d'une réflexion. Observation du phénomène de montagne. En haut: Plan M à gauche, vue sur l'axe XY à droite. En bas: vue 3D.

Pour déterminer le lobe spéculaire il faut partir de la zone d'intensité forte (le pic spéculaire), et inclure tous les pixels voisins de celle-ci jusqu'à ce que la variation

d'intensité entre deux pixels voisins soit faible. Cela revient à comparer les pixels voisins dans les deux directions horizontale et verticale, et descendre jusqu'à ce qu'on trouve une variation d'intensité faible. Saint-Pierre propose de comparer deux pixels voisins à un seuil (équation 2.8), seuil qui représente 1% de la variation de l'intensité dans l'image [10]. La valeur du seuil est donnée par l'équation 2.9, où le canal représente une trame (R, G, B, M, ou S); si la différence est inférieure à ce seuil alors on poursuit la descente.

$$Pixel_{actuel} - pixel_{suivant} > seuil \tag{2.8}$$

$$seuil = \frac{\max(canal) - \min(canal)}{100} \tag{2.9}$$

Bhattacharyya va plus loin et propose une deuxième approche [14]. Il propose un moyen de stopper la descente en intensité lorsqu'on a atteint les limites de l'objet. En procédant ainsi, il sera possible de limiter les dégâts dans la mesure où le pic détecté n'en est pas vraiment un. En effet, l'algorithme de détection peut confondre une zone naturellement blanche ou brillante à un pic. Stopper la descente en intensité lorsqu'on atteint les limites d'un objet évitera de considérer la région entourant cet objet naturellement blanc comme étant un lobe spéculaire. En plus, cela évitera de joindre involontairement deux régions voisines et donc de propager des erreurs de correction de l'une dans l'autre.

Malheureusement, ces méthodes ne peuvent pas facilement être appliquées pour un traitement en temps réel. La descente proposée par Bhattacharyya utilise une méthode appelée retinex afin de détecter les contours des régions dans l'image. Cette méthode

nécessite des calculs supplémentaires. Par contre le temps d'exécution nécessaire à la descente en intensité classique [10] variera en fonction de la taille du lobe spéculaire, et sera difficile à déterminer. Dans une architecture matérielle où les ressources à utiliser sont connues d'avance, il faudra déterminer de manière arbitraire une largeur maximale du lobe. De plus, cette descente demandera énormément de ressources matérielles. En effet, il faudra disposer de beaucoup de mémoire et ressources pour effectuer la boucle nécessaire à la descente dans les directions horizontale et verticale simultanément, surtout si le lobe est très large.

2.2 Correction des réflexions spéculaires

2.2.1 Description générale

Une fois le masque spéculaire créé, la prochaine étape est la correction des réflexions spéculaires. Plusieurs recherches ont été faites dans ce domaine, chacune utilisant des approches variées.

Kokaram et al. utilisent l'estimation du mouvement et des modèles autorégressifs afin d'interpoler les endroits à corriger à partir de trames adjacentes [15]. L'idée ici est de remplacer les pixels spéculaires par des pixels non spéculaires prélevés à la même position mais sur une trame voisine. L'avantage de cette méthode est sa simplicité. Il suffit de stocker en mémoire plusieurs trames, marquer les régions spéculaires, puis comparer leur contenu les unes avec les autres. Mais pour que cela fonctionne, il faudrait que les mouvements de la caméra soient assez lents afin que les pixels restent approximativement à la même position. Il faudrait aussi que la source lumineuse soit

indépendante de la caméra. De plus, cette méthode nécessite beaucoup de mémoire pour le stockage des trames.

Hirani et T. Totsuka proposent une méthode qui exploite le contenu fréquentiel et spatial de l'image [16]. Grâce au domaine fréquentiel les auteurs sont capables de traiter l'image dans sa globalité. En alliant le domaine spatial ils évitent l'inconvénient du traitement fréquentiel qui est la perte de certains détails tels que les contours. Leur méthode trouve sa limite dans la nécessité de connaissances à priori. En effet, l'utilisateur doit spécifier une sous-image ayant la même texture que la zone à corriger, ce qui pose déjà un frein pour une implémentation en temps réel. En plus de cela, la tâche deviendra très ardue s'il y a différentes textures dans la même image.

Lin propose une correction à partir de différentes vues de la même scène [17]. Cela consiste à remplacer les pixels spéculaires d'une vue par les pixels sains de l'autre. L'avantage par rapport à [15] est qu'on n'a pas besoin d'utiliser beaucoup de mémoire. Cependant, la méthode est efficace si la même scène est prise avec différents points de vue. Cela signifie avoir deux caméras, ce qui n'est pas usuel lors d'une chirurgie endoscopique.

2.2.2 La restauration d'image ou image inpainting

Les méthodes qui se prêtent bien aux vidéos endoscopiques et au traitement temps réel sont basées sur le principe de la restauration d'image. Ce sont des méthodes qui consistent à prélever des informations provenant des contours d'une zone à corriger et à les propager à l'intérieur de cette zone.

Il est possible d'exploiter le contexte de l'application pour améliorer et simplifier le processus de restauration d'image. Par exemple, St-Pierre a proposé des hypothèses spécifiques pour le cas d'images endoscopiques prises dans le corps humain [10]. Par exemple, on observe que pour un objet rond, les réflexions sont situées sur la partie centrale, donc à l'intérieur de celui-ci. Cela permet d'émettre l'hypothèse que la majorité des réflexions ont une bordure à partir de laquelle on peut extrapoler l'intérieur. Cependant, les méthodes basées sur la restauration d'image atteignent leurs limites lorsqu'elles sont confrontées à des objets plats, dont les réflexions spéculaires sont réparties sur la totalité de l'objet.

On observe que certains tissus dans le corps sont pâles : les os, quand ils ne sont pas recouverts de sang, et la compresse utilisée par le chirurgien. Ces tissus réfléchissent presque parfaitement la lumière, et peuvent donc être confondus à des réflexions spéculaires. On émet cependant l'hypothèse que la considération de certains tissus comme réflexion spéculaire ne nuit pas trop car la bordure de ces tissus est elle aussi pâle. Même en cas de correction, la couleur blanche sera propagée à l'intérieur des tissus, ce qui ne devrait pas trop dégrader le rendu.

Bertalmio propose une méthode qui consiste à reconstruire la zone à corriger sans modification des arêtes [18]. Les isophotes (lignes dont les pixels ont une intensité constante) arrivant à la bordure de la zone sont propagés à l'intérieur de celle-ci. La propagation est effectuée tout en préservant l'angle d'arrivée des isophotes au niveau des bordures. Considérons une image où Ω représente la zone à corriger, $\partial\Omega$ sa bordure, et $\vec{N}$ la normale à cette bordure (figure 2.10).

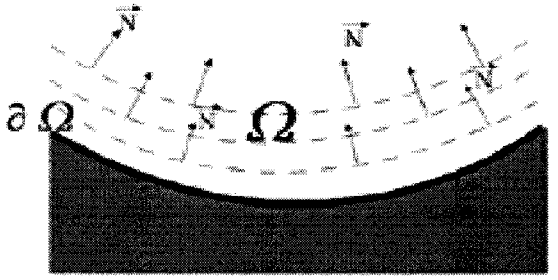


Figure 2-10 Zone à corriger ([18])

L'image corrigée I^{n+1} est trouvée à partir de l'image précédente I^n à laquelle on applique des informations globales I_t^n obtenues à partir de celle-ci ; cela est résumé par l'équation 2.10, où le couple (i, j) représente la position des pixels dans l'image

$$I^{n+1}(i,j) = I^n(i,j) + \Delta t.I_t^n(i,j), \forall (i,j) \in \Omega \qquad (2.10)$$

L'information I_t^n correspond à la variation de l'intensité projetée sur la normale à la bordure $\partial\Omega$. Le facteur Δt indique que la propagation est faite de manière périodique. Afin de conserver les contours des objets, une diffusion anisotropique est aussi effectuée périodiquement.

Cette méthode, bien que très efficace, peut se révéler instable à cause du nombre de paramètres arbitraires. Il faut en effet définir trois paramètres pour trouver I^{n+1} : le nombre d'itérations de restauration d'image k et de diffusions anisotropiques l à effectuer sur une période T. En plus deux critères d'arrêt de l'algorithme sont utilisés : la correction peut être effectuée un nombre n fois défini arbitrairement, ou encore jusqu'à ce que la différence entre I^{n+1} et I^n soit inférieur à un seuil défini arbitrairement en autant que le nombre de comparaison entre I^{n+1} et I^n est inférieur à n. Saint-Pierre a

montré que cette méthode peut être implémentée en temps réel [10]. Mais pour cela il ne faudrait pas être limité en mémoire.

Saint-Pierre propose une méthode assez simple à réaliser qui consiste à propager les arêtes sans tenir compte de l'information en termes d'intensité en bordure de la zone à corriger [10]. Cette méthode différente de [18] consiste à pondérer les pixels voisins du pixel à corriger de manière à déterminer leur influence dans la correction. La pondération est faite de telle sorte que la propagation soit perpendiculaire aux arêtes. Les avantages de cette méthode sont sa rapidité et sa stabilité. Mais ses résultats présentent des arêtes déformées et une non-continuité de l'image. Cela s'explique en partie par l'absence de diffusion anisotropique.

Une méthode encore plus simple a été proposée par Battacharya [14]. Elle consiste à remplacer la valeur du pixel spéculaire par la moyenne des pixels voisins non spéculaires. La correction est effectuée jusqu'à ce que la zone à corriger Ω disparaisse. Cette méthode qui pourrait très bien se porter à une implémentation en temps réel montre cependant une limite dans la détermination du critère d'arrêt. Celui-ci devrait tenir compte de la plus grande largeur à corriger dans l'image. Or nous n'avons pas cette connaissance à priori. Pour effectuer une bonne correction il faudrait considérer le pire cas, c'est-à-dire celui pour lequel la zone à corriger est répartie sur toute la trame. Une vidéo NTSC ayant 858 pixels sur une ligne, il faudrait passer à travers la même ligne 858 fois. En d'autre termes il faudrait passer à travers la même ligne une fois pour chaque pixel contenu dans une ligne. Cela signifierait un stockage de 858 lignes multipliées par les ressources utilisées pour le calcul de la moyenne des pixels. On aurait

donc $858x858x3 = 2.1Mo$ nécessaires pour traiter une trame d'une couleur si on considère qu'on a besoin de stocker 3 lignes pour effectuer la moyenne d'un pixel.

2.3 Conclusion

Les différentes approches présentées dans ce chapitre nous permettent d'avoir un aperçu de celles pouvant être implémentées dans notre système. Chacune ayant des avantages et des inconvénients, nous les combinons dans le chapitre suivant afin de créer des algorithmes efficaces pour une application temps réel tout en consommant peu de ressources.

Chapitre 3 Implémentation matérielle

Ce chapitre présente en détail les algorithmes choisis à partir des différentes méthodes énoncées dans le chapitre précédent. Il présente aussi la méthodologie utilisée pour effectuer l'implémentation de ces algorithmes sur un FPGA. La première partie est une présentation générale de l'environnement de prototypage et des caractéristiques de l'architecture d'un FPGA. La deuxième partie présente les détails de l'implémentation des deux algorithmes de détection. La dernière partie présente les détails de l'implémentation de l'algorithme de correction.

3.1 Présentation générale du système

3.1.1 Description de l'environnement de prototypage

Afin d'effectuer l'implémentation de notre processeur, nous essayons de simuler le plus possible les conditions réelles d'opération. Notre système de prototypage présenté en figure 3.1 comprend :

- La planchette de développement XUV2P de Digilent inc.
- Un écran VGA relié à la planchette de développement
- Un lecteur DVD standard
- Un DVD contenant des séquences vidéos endoscopiques
- Un décodeur vidéo

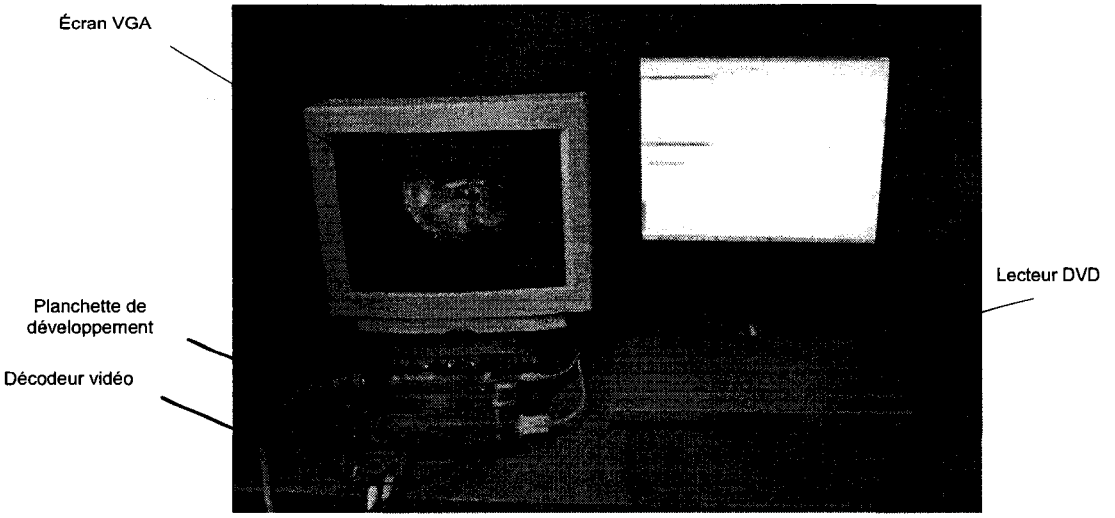


Figure 3-1 Environnement de prototypage

La planchette de développement présentée en figure 3.2 est bâtie autour du FPGA XC2VP30. Le tableau 3.1 présente les principales caractéristiques de ce FPGA.

Tableau 3-1 Ressources disponibles du FPGA XC2VP30

Composants	**Ressources disponibles**
Slices	13969
Ram distribuée	428 Kb
Blocs multiplicateurs	136
Blocs RAMs	2448 Kb

La planchette est aussi munie d'un port d'expansion auquel est connecté le décodeur vidéo ADV7183B de la compagnie Analog Devices. Ce décodeur détecte et convertit

automatiquement un signal analogique vidéo standard de type NTSC, PAL ou SECAM en signal numérique.

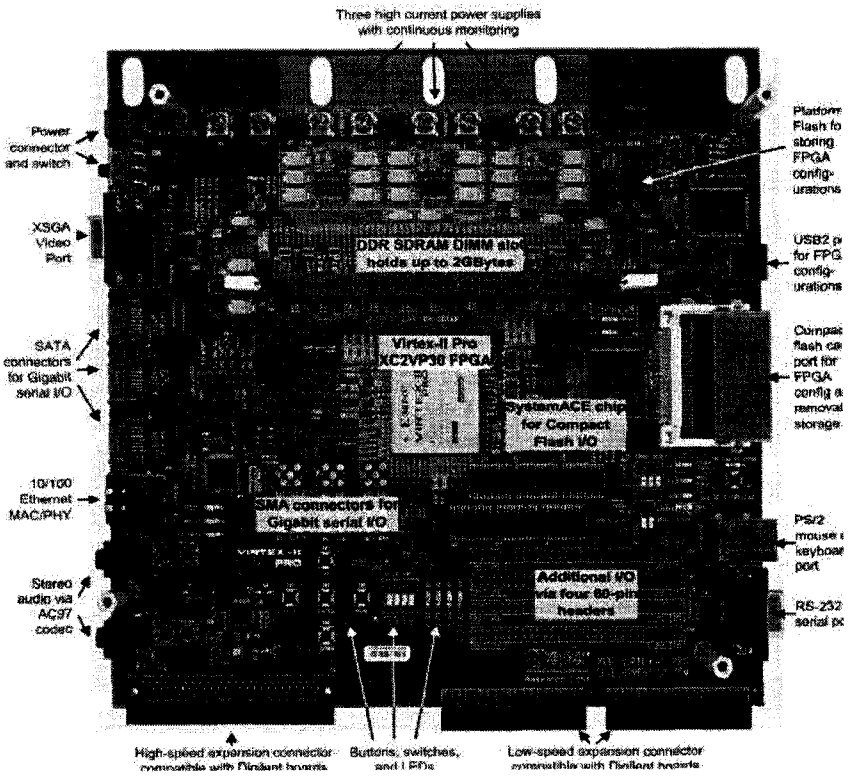


Figure 3-2 Planchette de développement XUV2P (www.digilentinc.com)

Il est muni pour cela de trois types d'entrées : composite, S-vidéo et component. Une de ces entrées est reliée au lecteur DVD afin d'acquérir les séquences vidéo endoscopiques du DVD. Ces séquences vidéos ont été préalablement prises lors d'une séance d'opération. Elles sont gravées sur le DVD afin de simuler le plus possible une situation réelle d'opération par endoscopie. Elles permettent dont d'effectuer un prototypage sans avoir besoin de se rendre dans une salle d'opération. Cependant les séquences vidéos ayant une résolution d'origine de 352x240 sont compressées afin d'être stockées sur le

DVD ; le lecteur DVD fait ensuite une mise à l'échelle afin de les afficher au format NTSC 720x480. Ces deux opérations sont susceptibles de dégrader le rendu, et nous éloigne des conditions réelles d'opération.

Le décodeur vidéo ADV7183B transforme les signaux analogiques en signal numérique de type YCrCb 4:2:2 entrelacé. Une fois acquis par le FPGA, ce signal passe par un bloc de désentrelacement qui fournit à notre système de traitement un signal NTSC progressif ayant une résolution de 720x480 à une fréquence de 27 MHz.

3.1.2 Signaux d'entrée du système de traitement

Commençons d'abord par préciser les principaux signaux en entrée : il s'agit des signaux R,G,B pour la valeur de chaque pixel, et des signaux de synchronisation. Chaque signal de couleur est donné sur 8 bits, ce qui donne un total de 24 bits de données. Les signaux de synchronisation permettent de guider les électrodes responsables du balayage vertical et horizontal présentes dans un écran à tube cathodique. Ces signaux indiquent quel type de balayage est en train d'être exécuté et la position du pixel actuel. Ils sont reçus en entrée du système puis retransmis après traitement au port VGA en vue d'un affichage sur un écran VGA. Globalement ces signaux permettent de diriger les électrodes de l'écran pour pouvoir effectuer un balayage horizontal de gauche à droite (Hsync pour Horizontal synchronization) et un balayage vertical de haut en bas (Vsync pour Vertical synchronization). Hsync permet de revenir à la ligne tandis que Vsync permet de remonter au début de l'écran. La figure 3.3 présente l'évolution temporelle de Hsync et Vsync pour un signal NTSC présentant

858 pixels par ligne, pour un total de 525 lignes. FrontPorch et BackPorch représentent respectivement les parties droite et gauche de l'affichage dans le cas de la synchronisation horizontale, et les parties inférieure et supérieure de l'affichage dans le cas de la synchronisation verticale.

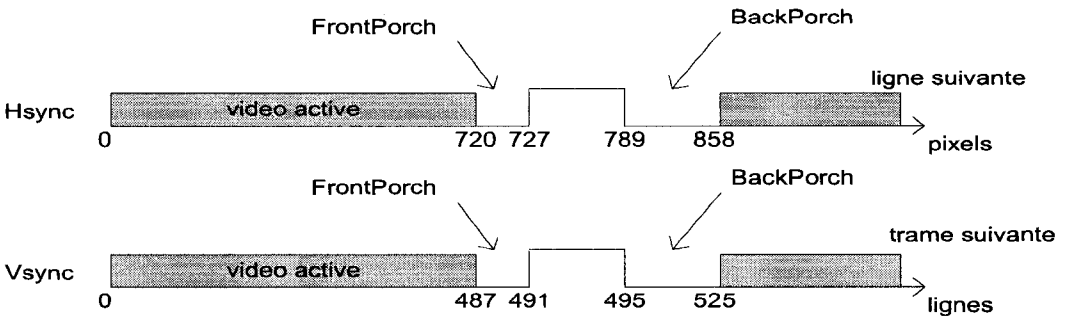


Figure 3-3 Evolution temporelle de Hsync et Vsync

On observe un temps pendant lequel rien n'est affiché sur l'écran. Celui ci correspond au temps nécessaire à la remise à la ligne ou au temps nécessaire à la remise au début de l'écran ; durant ce temps le système est en mode « video inactive ». C'est pendant cette période que la plupart des opérations peuvent être effectuées de manière transparente. La durée de ce mode est de 38 lignes soit 32604 coups d'horloge ou pixels pour une résolution de 720x480 (plus précisément, 720x487) à une fréquence de 27 MHz.

3.2 Contraintes liées à l'utilisation d'un FPGA

3.2.1 Parallélismes au sein d'un FPGA

L'utilisation d'un FPGA pour le traitement vidéo à la place d'un processeur d'usage général se justifie par la possibilité d'exploiter deux types de parallélisme au sein d'un FPGA, contrairement au traitement séquentiel du processeur d'usage général [20].

Le premier parallélisme possible est le parallélisme spatial. Il consiste à diviser la trame en sous-trames traitées de manière concurrente chacune par un processeur (les processeurs sont donc tous identiques et répliqués), puis recomposées en sortie. Ceci nécessite d'avoir au préalable stocké la trame initiale en mémoire avant de faire le traitement. Bien que l'augmentation du nombre de processeurs augmente la vitesse de traitement, cela provoque aussi un effet de recouvrement aux bordures des divisions.

Il est aussi possible d'effectuer un parallélisme temporel. En effet, puisque différentes opérations doivent généralement être effectuées sur une trame, il est possible de les répartir par processeur et de les exécuter de manière concurrente, c'est-à-dire les effectuer en même temps mais sur des trames différentes. Par exemple, une fois que le premier processeur a fini de traiter la trame 1, il la passe au processeur 2. Pendant que le processeur 2 traite la trame 1, le processeur 1 traite la trame 2. Le problème à ce niveau est que la vitesse globale d'exécution est déterminée par le processeur le plus lent. Dans le cas d'un système NTSC fonctionnant en temps réel, le processeur le plus lent devrait fonctionner à un minimum de 27 MHz.

Ces deux parallélismes peuvent être utilisés simultanément en divisant d'abord la trame reçue en sous-trames puis en effectuant du parallélisme temporel sur chacune d'elles. L'architecture que nous utilisons ressemble plus à du parallélisme temporel car chaque trame est traitée entièrement par un processeur et non divisée en sous-trames. Toutefois, nous avons deux processeurs qui traitent la même trame, le deuxième se servant de l'information du premier lors du traitement de la trame précédente.

3.2.2 Modes de traitements de données dans un FPGA

Programmer sur FPGA diffère grandement d'une programmation sur un microprocesseur. Non seulement on implémente l'algorithme, mais aussi son architecture. Notre système comprend plusieurs processeurs qui travailleront tous en parallèle, certains devant accéder aux mêmes ressources simultanément. Il est donc par exemple nécessaire de tenir compte de la bande passante limitée, surtout lors des accès en mémoire. Il existe trois modes de traitements de données, chacun ayant son lot de contraintes, que nous présentons ci-dessous [21].

Le mode « flux » : ici les données sont présentées en entrée du système comme un flux de données reçues à une fréquence d'horloge fixe. Pour une vidéo NTSC elles sont reçues à une fréquence de 27 MHz, et doivent aussi être émises à la même fréquence. Cette restriction entraîne des contraintes temporelles rigides surtout dans le cas d'un traitement en temps réel. En effet, le mode flux contraint le système à effectuer toutes les opérations requises pour chaque pixel à une fréquence de 27 MHz. Si le processeur implémenté a une fréquence de 100 MHz, il ne pourra donc effectuer que 3 opérations

par pixels. Si cela n'est pas possible, certains pixels dans le flux seront manqués et ne seront pas calculés. Dans le cas où des opérations complexes doivent être effectuées sur les pixels, il est impossible de respecter les contraintes de temps sans effectuer du « pipelinage ». Cela consiste à créer un délai en entrée afin de mettre en place les différents calculs. En d'autres termes, cela revient à créer un parallélisme temporel. Étant donné qu'on obtiendra la même latence en sortie, il faudrait s'assurer que celle-ci n'est pas gênante pour l'utilisateur. De plus les contraintes liées aux ressources disponibles apparaissent ici, car effectuer du pipelinage reviendrait à utiliser des registres afin de stocker temporairement les données avant leur traitement.

Un mode très souple est le mode « hors-ligne ». Dans ce cas les données traitées proviennent d'une image au préalable stockée dans une RAM. La vitesse d'exécution est alors limitée par la vitesse d'accès aux données stockées dans la mémoire. Ici il n'y a donc pas de contrainte de bande passantes, mais le temps de calcul est très grand. En plus les ressources utilisées sur le FPGA en lui-même sont minimes car il n'est pas nécessaire d'effectuer du pipelinage ou des stockages de trames ou partie de trame sur la mémoire disponible sur puce. Pour se prêter au temps réel ce mode devrait avoir une vitesse de traitement assez grande pour avoir le temps d'effectuer le traitement sur une trame avant de charger la suivante. Le mode hors ligne convient très bien pour un parallélisme spatial pur.

Le troisième mode est un mixage entre les deux méthodes ci-dessus. Il s'agit du mode « hybride ». Il permet d'alléger les contraintes de temps liées au mode « flux » tout en s'appliquant en temps réel. En effet il est possible, si les ressources disponibles le

permettent, de stocker les différents pixels arrivant dans une RAM, de les traiter ensuite en mode hors-ligne, puis de les ressortir en mode flux après un délai constant. En plus des contraintes de temps, il s'ajoute des contraintes liées aux ressources du système. Pour une trame couleur de 720x480 pixels, il faudrait près de 720x480x3x8 bits, soit environ 1Mo, si on désire stocker les trames rouge, verte et bleue ayant chacune des pixels pouvant avoir 255 valeurs différentes.

Les ressources en mémoire interne du FPGA étant très limitées, on fait généralement appel à une mémoire externe. Celle-ci entraîne une contrainte liée à son temps d'accès qui doit être inférieur au temps de réception d'un nouveau pixel. De plus les mémoires externes généralement utilisées ne permettent pas un accès multiport, c'est-à-dire un accès à différentes adresses au même moment. Il faudrait donc aussi tenir compte de cette contrainte de bande passante.

Malgré ses contraintes temporelles, nous utilisons le mode flux dans le cadre de la détection de réflexions spéculaires ; nous le préférons au mode hybride pour éviter des problèmes de synchronisation avec la RAM externe, et pour pouvoir diminuer le temps de latence entre la réception d'un pixel et la sortie de celui-ci une fois traité. Les détails de l'implémentation seront donnés plus bas.

3.3 Architecture globale du système

Deux méthodes de détection ressortent du chapitre deux, la méthode monodimensionnelle et la méthode bidimensionnelle. Chacune nécessitant une implémentation particulière, nous obtenons deux architectures globales.

D'un point de vue général, les deux architectures utilisées pour la détection et la correction des réflexions spéculaires sont similaires. Afin d'exploiter le parallélisme inhérent aux FPGAs, nous créons deux tâches qui s'exécutent en parallèle. Comme spécifié dans la section 3.2, le parallélisme utilisé ici ressemble à du parallélisme temporel, deux tâches s'exécutant de manière concurrente et traitant la même trame en entrée. Cependant, l'une se sert de l'information de l'autre pour effectuer son traitement. En effet la tâche 2 se sert de la valeur du seuil déterminée dans la tâche 1 afin d'effectuer la correction.

Les signaux de synchronisation et les signaux de couleur RGB provenant du décodeur vidéo sont tout d'abord désentrelacés avant d'entrer dans le système (figure 3.4).

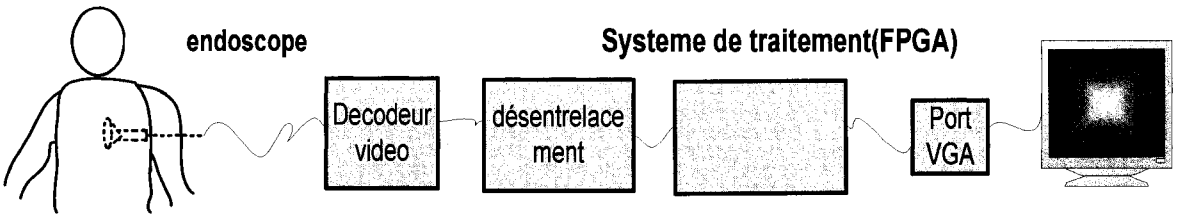


Figure 3-4 Schéma global du système

Pour le système utilisant une détection monodimensionnelle (figure 3.5, blocs chemin en traits verts), les signaux passent d'abord par le bloc de décomposition en histogramme. L'histogramme créé est stocké dans une RAM. Une fois que le système passe en mode de vidéo inactive, le bloc de débruitage est activé (grâce au signal *fini*) afin de débruiter l'histogramme stocké. L'histogramme débruité passe ensuite dans le bloc de dérivation, afin de déterminer le début de la bosse spéculaire.

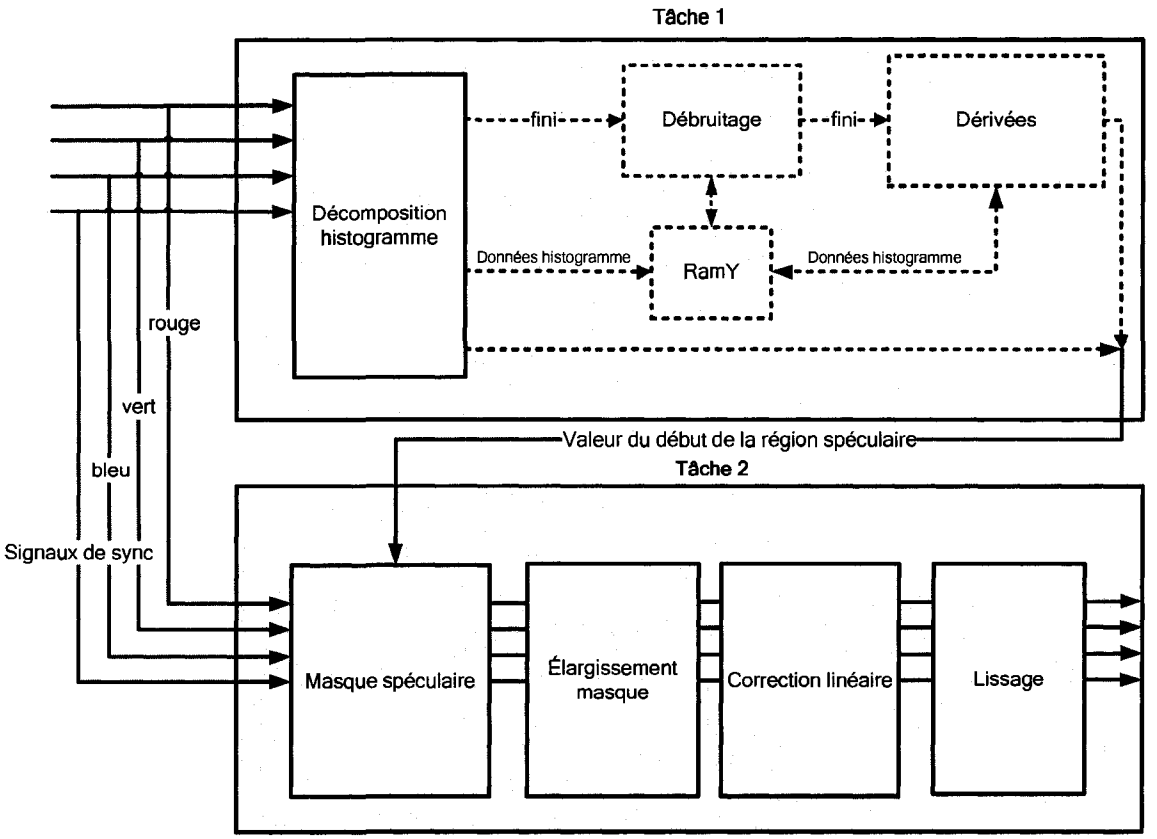


Figure 3-5 Schéma bloc du système ; en vert : détection monodimensionnelle ; en turquoise : chemin détection bidimensionnelle

Pour le système utilisant une détection bidimensionnelle (figure 3.5, chemin en trait turquois), les signaux passent par un bloc de décomposition en histogramme ; contrairement à l'autre système, aucun stockage n'est effectué. La décomposition en histogramme et la mise à jour du seuil sont effectuées simultanément. En sortie de ce bloc, nous obtenons donc le seuil.

Dans les deux systèmes, la valeur du seuil est utilisée conjointement avec les signaux de couleur et de synchronisation afin de créer le masque spéculaire. Celui-ci est ensuite élargi afin de tenir compte du lobe spéculaire. Les pixels d'origine (les signaux de

couleur présents à l'entrée du système) et les pixels du masque spéculaires sont transmis au bloc de correction linéaire puis de lissage afin que l'image soit corrigée. Les pixels corrigés, ainsi que les signaux de synchronisation ayant un délai égal au délai du traitement de la tâche 2, sont mis en sortie et transmis au port VGA afin d'être affichés sur un écran (figure 3.5).

Les deux architectures utilisent des données entières à 8 bits pour stocker les différentes valeurs liées aux pixels et relier les différents blocs de traitement.

3.4 Détection des réflexions spéculaires

3.4.1 Méthode unidimensionnelle

L'algorithme implémenté se sert des principes énoncés dans le chapitre 2. Afin de garantir une implémentation matérielle, nous modifions les équations théoriques tout en conservant une erreur minimale entre les résultats théoriques et les résultats obtenus après implémentation.

La méthode unidimensionnelle se sert d'une RAM pour stocker l'histogramme. Celui-ci est ensuite traité lorsque l'écran entre en mode vidéo inactive afin d'en déduire le début de bosse spéculaire

3.4.1.1 Décomposition en histogramme

Nous proposons une approche basée sur des méthodes existantes présentées en section 2.1.1.1: nous rehaussons l'image grâce à la composante S, puis transformons l'image rehaussée en niveaux de gris avant de la décomposer en histogramme. La décomposition

en niveau de gris permet de stocker toute l'information importante dans un seul canal, et donc de travailler sur un seul histogramme, comme celui présenté en figure 3.6. On peut remarquer que le rehaussement permet de mieux regrouper tous les pixels appartenant à la région diffuse ; cela a pour effet de mettre plus en évidence la zone spéculaire. L'utilisation d'un seul histogramme permet de limiter grandement l'utilisation des ressources du FPGA car nous créons et traitons une seule RAM au lieu de 3.

La décomposition en histogramme s'effectue en plusieurs étapes. A chaque fois qu'un pixel est reçu :

- Il est rehaussé
- Il est transformé en niveau de gris
- Il est utilisé pour mettre à jour l'histogramme

Le rehaussement est effectué grâce à l'équation 3.1, où R'G'B' représentent les valeurs rehaussées de RGB.

Cette équation devant être appliquée sur un système qui n'utilise que des données entières, il convient d'effectuer des modifications afin d'avoir une bonne approximation des valeurs réelles. Nous la transformons pour cela en l'équation 3.2.

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = (1-S)\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \frac{\min(R,G,B)}{\max(R,G,B)}\begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.1}$$

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \frac{(\frac{256}{\max(R,G,B)}) \times \min(R,G,B)}{256} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.2}$$

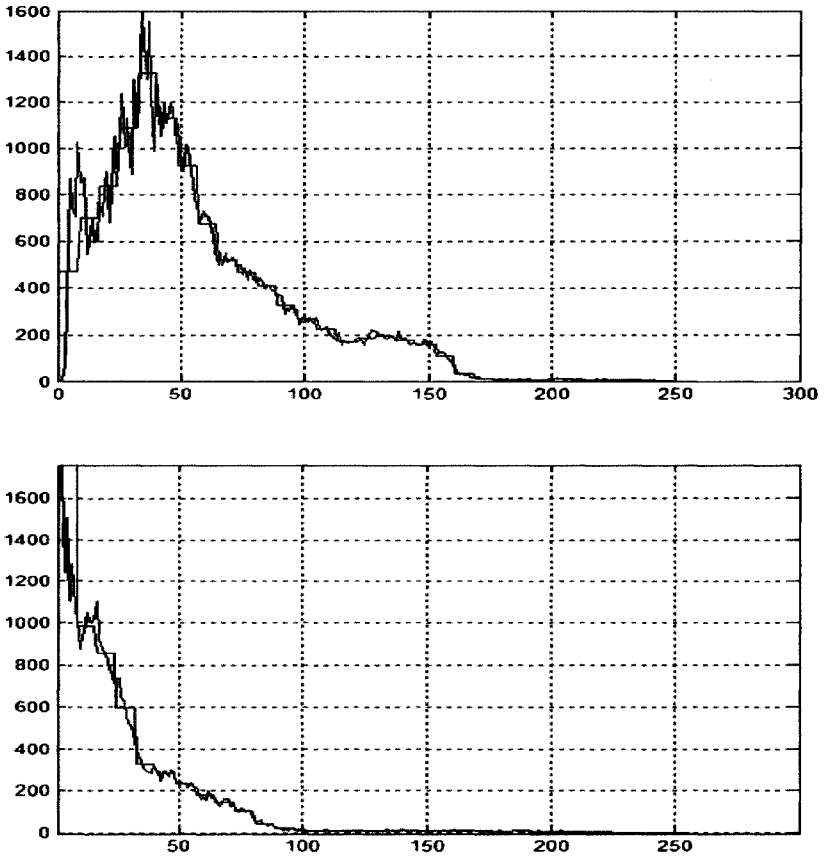


Figure 3-6 Histogrammes de l'image en figure 2.1 : histogrammes du plan Y sans rehaussement(en haut) et avec rehaussement (en bas). En bleu : histogramme non débruité ; en noir : histogramme débruité

L'équation 3.2 permet d'éviter le processus de division $\frac{\min(R,G,B)}{\max(R,G,B)}$, processus qui n'est pas directement synthétisable, à moins d'implémenter un module spécial pour la division. À la place, nous créons et stockons un tableau de valeurs entières $\frac{256}{i}$. Le rehaussement s'effectuera donc en déterminant premièrement la plus grande valeur entre les différentes intensités RGB d'un pixel donné. Cette valeur servira d'indice pour accéder au tableau stocké présenté dans le tableau 3.2. Afin d'éviter un cas de division

par 0, nous attribuons la valeur nulle au résultat $\frac{256}{i}$ lorsque $i=0$. Le résultat attribué n'a pas une grande importance car si $i=0$, alors max(R,G,B)=min(R,G,B)=0.

Tableau 3-2 Tableau des valeurs

i	0	1	2	3	5	...	255
256/i	0	256	128	85	64		1

Une fois tout le processus de multiplication effectué, la valeur sera décalée à droite de 8 bits afin d'effectuer la division par 256.

Deux approximations sont effectuées ici. Premièrement, la création d'un tableau nécessite un arrondi des valeurs décimales en valeurs entières. Cela crée une erreur maximale e_1 de ± 0.5, erreur calculée en effectuant la différence entre les valeurs arrondies et les valeurs réelles du tableau. Cela est montré par l'équation 3.3 où $\text{int}(\frac{256}{i})$ représente la partie entière de la division. Deuxièmement, la division par 256 nécessite elle aussi un arrondi en valeurs entières. L'erreur maximale e_2 est aussi de ± 0.5, et provient du calcul de l'équation 3.4, où i varie de 0 à 255^2, valeur maximale du numérateur de 3.2.

$$e_1 = \max\left[abs((\frac{256}{i}) - \text{int}(\frac{256}{i})) \right] i \in [0;255] \tag{3.3}$$

$$e_2 = \max\left[abs((\frac{i}{256}) - \text{int}(\frac{i}{256})) \right] i \in [0;255^2] \tag{3.4}$$

Ces deux approximations créent une erreur maximale de ± 1 lorsque le rehaussement est effectué.

La transformation en niveaux de gris s'effectue juste après le rehaussement. Elle est faite à l'aide de l'équation 2.3 mise en une forme synthétisable. La forme synthétisable permet aussi ici d'arrondir les valeurs décimales en valeurs entières. Elle est donnée par l'équation 3.5

$$Y = \frac{[0.299 \times 512] \times R + [0.587 \times 512] \times G + [0.114 \times 512] \times B}{512}$$

$$Y = \frac{153R + 301G + 58B}{512} \tag{3.5}$$

En appliquant le même processus de vérification que dans le cas du rehaussement, on obtient une erreur maximale de ± 1.

Le rehaussement et la transformation en niveaux de gris sont des calculs purement combinatoires qui s'effectuent à l'intérieur d'un seul cycle d'horloge. Les valeurs obtenues permettent la construction d'un histogramme. La construction et la mise à jour de la RAM contenant l'histogramme n'est pas aisée car elle nécessite de tenir compte de certaines considérations.

Premièrement, nous devons tenir compte de sa taille. Le nombre maximum de pixels pouvant être compris dans une plage d'intensité est 720x487=350640 pixels dans le cas où toute la trame a la même couleur. Ce nombre étant représentable avec 19 bits, la RAM devrait avoir pour taille 256x19bits. Cela signifierait travailler continuellement avec des données de 19 bits lors du traitement de la RAM. Pour éviter cela, nous supposons que la partie utile (partie où se situe la région spéculaire) de l'histogramme se

situe après la première région de l'histogramme, région dans laquelle est située la majorité des pixels. Nous supposons aussi que dans la seconde région, les différentes plages d'intensité n'ont pas plus de 256 pixels. Nous utilisons une RAM de 256x8 bits où les valeurs supérieures à 255 seront seuillées à 255, ce qui nous permet de travailler de manière uniforme avec des données de 8 bits tout au long du processus de détection des réflexions spéculaires.

Puisque nous travaillons en mode «flux » nous devons continuellement accéder à la RAM, à chaque pixel reçu. La mise à jour se fait en trois étapes : la lecture de la RAM à partir d'une adresse correspondant à la valeur en nuances de gris, la réception de la donnée provenant de la RAM, et la mise à jour de cette valeur dans l'histogramme. Afin de garantir une mise à jour constante de la RAM, nous utilisons une horloge qui fonctionne 4 fois plus vite que l'horloge de réception des pixels, c'est-à-dire 108 MHz. Cela permet de mettre à jour l'histogramme avant que le prochain pixel ne soit reçu. Le module permettant la mise à jour de l'histogramme est présenté en figure 3.7.

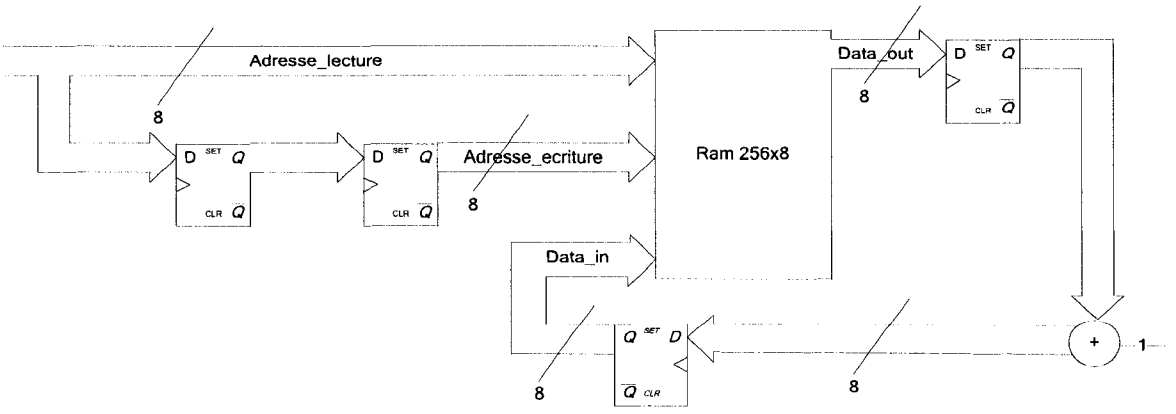


Figure 3-7 Processus de mise à jour de la Ram contenant l'histogramme

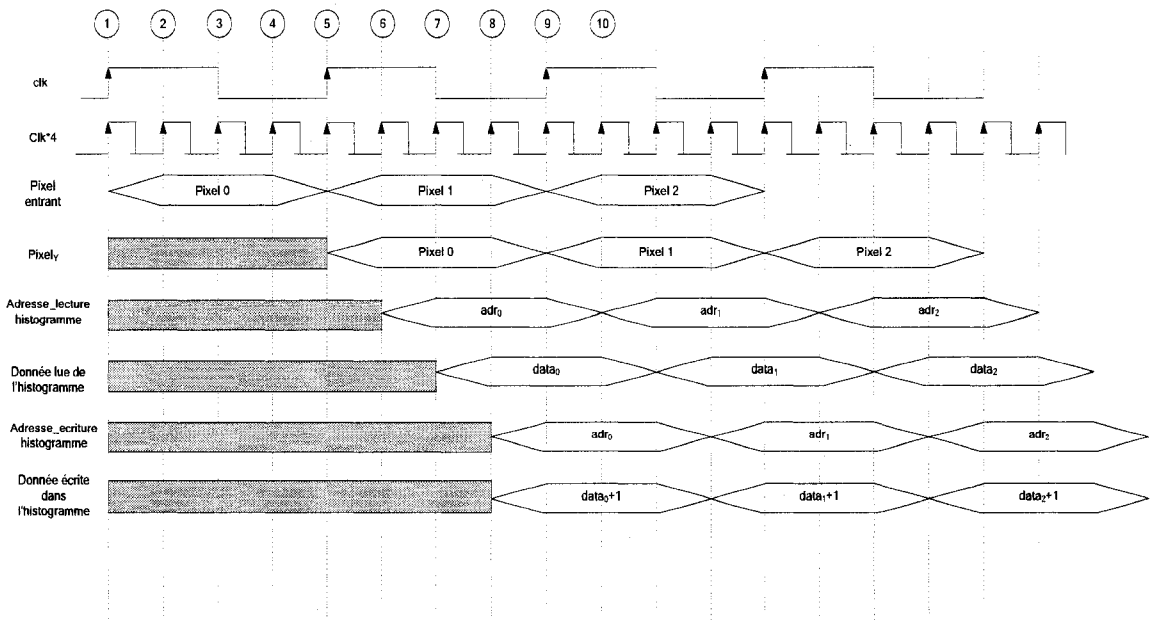


Figure 3-8 Diagramme temporel présentant la décomposition en histogramme

Tout se passe comme s'il y avait en sortie de la RAM une bascule D dont la sortie est reliée à un additionneur. Chaque valeur sortie par la RAM est donc registrée, incrémentée de 1, puis remise en entrée de la RAM afin d'être sauvegardée. Le fonctionnement global de la décomposition en histogramme est donné par le diagramme temporel présenté en figure 3.8 :

- Au premier coup d'horloge de 27MHz (clk), un pixel p_o est reçu.
- Au second coup d'horloge de 27 MHz (clk), la valeur en Y de p_o est obtenue. Le pixel suivant p_o est reçu.
- Au 6^e coup d'horloge de 108 MHz (clk*4), la valeur en Y de p_o est mise en adresse de lecture de la RAM. Cette valeur en Y est encore appelée adr_0.
- Au 7^e coup d'horloge de clk*4, la donnée stockée à l'adresse adr_0 est reçue. Cette donnée est nommée $data_0$.

- Au 8^{e} coup d'horloge de clk/4, la donnée $data_0$ est incrémentée de 1 et écrite à l'adresse adr_0.

Le délai total permettant la décomposition en histogramme est au final de un cycle d'horloge, temps nécessaire au calcul la valeur Y d'un pixel.

3.4.1.2 Débruitage de l'histogramme

Le débruitage s'effectue à la suite de la décomposition en histogramme. Il s'active lorsque le système entre en mode vidéo inactive. Cela permet d'accomplir le traitement de l'histogramme sans perte de données, la RAM ne pouvant pas être mise à jour en même temps.

Le débruitage s'effectue grâce l'ondelette de Haar, la plus simple des ondelettes [24]. Afin de décomposer le signal en coefficients d'approximation et de détail, on se sert des éléments voisins du signal deux à deux. Les coefficients d'approximation a et de détail d d'un signal s sont obtenus grâce aux équations 3.6 et 3.7 respectivement :

$$a(i) = \frac{s(2i) + s(2i+1)}{2} \tag{3.6}$$

$$d(i) = \frac{s(2i) - s(2i+1)}{2} \tag{3.7}$$

En d'autres termes, pour deux pixels du signal d'origine, un coefficient d'approximation et un coefficient de détail sont calculés (figure 3.9).

Deux tableaux sont créés, l'un pour stocker les coefficients d'approximation, l'autre pour les coefficients de détail. Lors du premier niveau de la transformation, les coefficients sont stockés dans la première moitié du tableau, première moitié qui

permettra d'effectuer la transformation de niveau 2. Les coefficients de niveau 2 seront à leur tour stockés à la suite des coefficients de niveau 1 et permettront le calcul des coefficients de niveau 3 (figure 3.10).

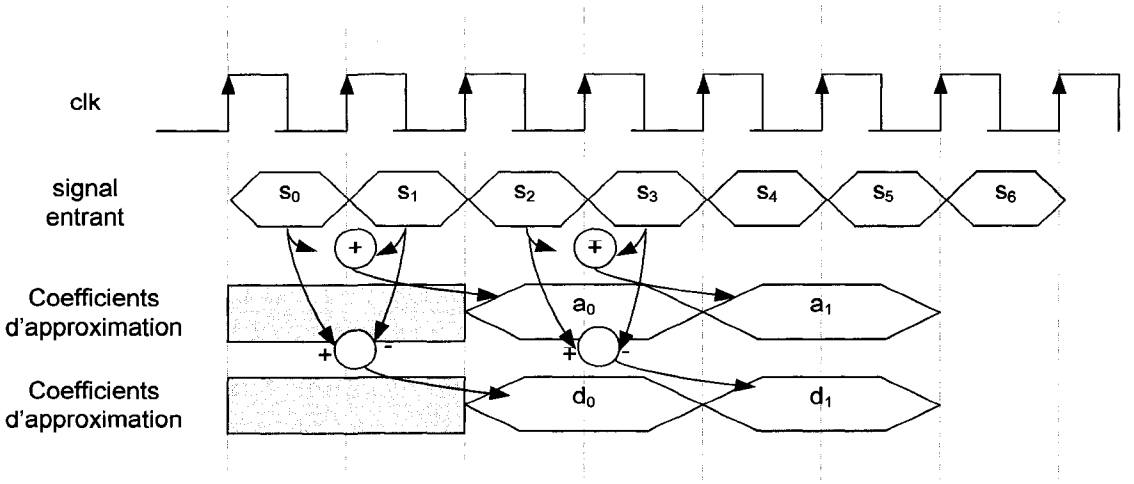


Figure 3-9 Décomposition du signal par l'ondelette de Haar

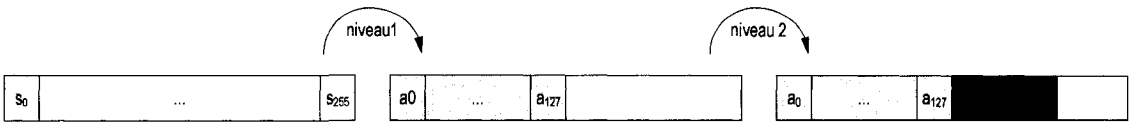


Figure 3-10 Stockage des coefficients de la transformation en ondelettes

La recomposition est implémentée en utilisant l'équation 3.8.

$$\begin{cases} s(2i) = approx(i) + détail(i) \\ s(2i+1) = approx(i) - détail(i) \end{cases} \tag{3.8}$$

Elle devrait s'effectuer à la suite d'un seuillage des coefficients des détails. Après plusieurs tests sur Matlab nous nous sommes rendu compte qu'une décomposition en ondelettes de niveau 3, puis recomposition sans tenir compte des coefficients de détails,

donnait des résultats satisfaisants. La figure 3.6 présente l'histogramme débruité de notre image test. On se rend compte que seules les grandes variations de la courbe sont conservées après débruitage.

Nous ne considérons pas les coefficients de détail pour notre recomposition, ce qui revient à répliquer 4 fois chacun des coefficients d'approximation de niveau 3 afin d'obtenir le signal débruité (figure 3.11).

Figure 3-11 Recomposition du signal après débruitage

Le temps de calcul pour le débruitage d'un signal est de 676 coups d'horloge car chaque coefficient est déterminé en un cycle d'horloge. Ce temps de calcul provient de la décomposition (256+128+64) suivi de la reconstruction (32+64+128). En effet il faut 256, 128 et 64 cycles d'horloges pour les décompositions de niveau 1, 2 et 3 respectivement. Il faut ensuite 32, 64 et 128 cycles d'horloges pour les reconstructions de niveau 3, 2 et 1 respectivement. À ce temps de calcul on ajoute le délai correspondant à l'accès à la RAM (la RAM étant registrée en entrée et en sortie, ce délai est de 3 cycles d'horloge) et le temps de calcul de la première valeur (1 cycle d'horloge).

3.4.1.3 Calcul de la dérivée

Deux dérivées s'effectuent l'une à la suite de l'autre. Elles se font en utilisant la formule de la différence centrée (équation 3.9) afin de déterminer la dérivée, où $s'(i)$ représente

la dérivée du pixel $s(i)$. Lorsque $h=1$, on obtient la formule finale 3.10. Cette formule revient à convoluer le signal à dériver par le noyau défini à la figure 3.12. La figure 3.13 quant à elle présente un exemple de convolution.

$$s'(i) = \frac{s(i+h) - s(i-h)}{2h} \tag{3.9}$$

$$s'(i) = \frac{s(i+1) - s(i-1)}{2} = s(i) \otimes (\delta(i-1) + \delta(i+1)) \tag{3.10}$$

-1/2	0	1/2

Figure 3-12 Noyau de convolution pour la dérivation

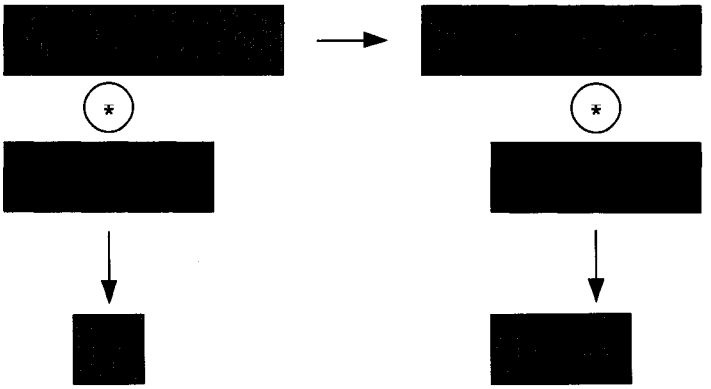


Figure 3-13 Deux étapes de dérivation

Toutes les valeurs positives sont mises à 1 et les résultats négatifs sont mis à 0. Le temps de traitement est de 261 cycles d'horloge, dont 256 correspondent au nombre de valeurs de l'histogramme et 5 cycles le délai entre la demande d'accès à la première valeur de l'histogramme et la sortie de la première valeur de la dérivée. Il tient donc en compte le temps d'accès à la RAM en lecture (3 cycles d'horloge) et en écriture (1 cycle

d'horloge), puis le temps de calcul (1 cycle d'horloge). Deux dérivées étant effectuées l'une à la suite de l'autre, nous avons un temps de traitement de 522 cycles d'horloge. Lors du calcul de la seconde dérivée, la dernière valeur positive est mise en sortie. Elle correspondra à l'intensité minimale d'un pic spéculaire, soit le début de bosse spéculaire.

3.4.1.4 Affichage du masque

Le module d'affichage du masque permet la création du masque binaire qui indique la position des pics spéculaires. Pour ce faire, il se sert du début de bosse spéculaire trouvé après la seconde dérivée. Idéalement, ce module devrait se situer à la suite de la seconde dérivation. Cela nécessiterait un stockage de la trame de départ (trame représentant l'image dans le plan RGB), puis une comparaison des valeurs au début de bosse afin de déterminer les zones de réflexion spéculaire. Pour éviter un stockage de la trame initiale, nous exécutons la décomposition en histogramme et l'affichage du masque en parallèle tel que représenté par la figure 3.5. En d'autres termes, nous utilisons les résultats de la trame i pour déterminer le masque de la trame i+1. Pour utiliser cette méthode, nous émettons l'hypothèse que les zones de réflexion spéculaire ont une variation négligeable entre deux trames successives. Cette hypothèse est vraie uniquement lorsqu'il n'y a pas de changement soudain de direction de la caméra. Dans tous les cas, l'erreur sera limitée à une seule trame d'une durée inférieure à 17 ms.

Hormis la phase de mise à jour de l'histogramme, ce module fonctionne de la même manière que celui de la décomposition en histogramme. Il effectue un rehaussement,

puis une transformation en niveaux de gris. Ensuite chaque valeur transformée est comparée au début de bosse spéculaire trouvé lors de la seconde dérivation de l'histogramme de la trame précédente.

3.4.2 Méthode bidimensionnelle

3.4.2.1 Décomposition en diagramme

Comme expliqué dans le chapitre 2, la décomposition en diagramme MS consiste à trouver le plan S en fonction du plan M. Il faudrait pour cela trouver le plan S, le stocker dans une RAM, puis y effectuer les calculs nécessaires à la détection de la zone spéculaire, calculs qui passent par la recherche de m_{max} et s_{max}. Afin de limiter l'utilisation des ressources, nous procédons de la manière suivante : pour chaque pixel recu,

- Sa valeur dans le plan M est calculée
- À partir de sa valeur dans le plan M, sa valeur dans le plan S est calculée
- Les valeurs m_{max} et s_{max} de la trame sont mises à jour

L'équation utilisée pour calculer le plan M provient de l'équation 2.5. Elle est cependant modifiée afin d'être synthétisable tout en utilisant peu de ressources. L'équation utilisée est donc l'équation 3.11.

$$M = \frac{\frac{256}{3} \times (R+G+B)}{256} = \frac{85 \times (R+G+B)}{256} \tag{3.11}$$

La division par 3 qui n'est pas directement synthétisable, à moins d'implémenter un module spécial pour la division, est remplacée par une multiplication suivie d'un

décalage à droite de 8 bits. L'erreur maximale créée par cette modification est calculée grâce à l'équation 3.12. Elle a une valeur de ± 2 .

$$e_3 = \max\left[abs(\frac{i}{3} - \text{int}(\frac{85 \times i}{256})) \right], i \in [0:767] \tag{3.12}$$

L'équation utilisée pour calculer la valeur du pixel dans le plan S demeure l'équation 2.6 car une division par 2 est parfaitement synthétisable (elle correspond à un décalage à droite d'un bit). L'erreur maximale due à la considération de la partie entière du résultat est de ± 1 .

Mettre à jour m_{max} et s_{max} régulièrement permet d'obtenir au terme du parcours de la trame les valeurs nécessaires à la détection du masque. Ces valeurs sont envoyées au module responsable de l'affichage du masque.

Le délai total de la décomposition est de un cycle d'horloge, les calculs étant tous purement combinatoires.

3.4.2.2 Affichage du masque

Une fois le diagramme MS établit, nous devrions y effectuer un seuillage basé sur les équations 2.7. Malheureusement, ces équations ne donnent pas de bons résultats dans le cas d'images endoscopiques car elles ont tendance à détecter uniquement les pixels extrêmement brillants. Pour obtenir une zone adaptée à notre cas, nous avons fait plusieurs tests et avons trouvé nos propres équations. Le pixel p sera considéré comme spéculaire s'il respecte les conditions suivantes :

$$\begin{cases} m_p \geq \frac{1}{2} m_{\max} \\ s_p \leq \frac{1}{3} s_{\max} \end{cases} \tag{3.13}$$

La figure 3.14 présente le diagramme MS correspondant à l'image de la figure 2.1 ; on constate que le choix de c_3 et c_4 tels que définis par (2.7) comme seuils ne permettrait pas de retrouver toutes les réflexions spéculaires de l'image ; en effet, très peu de pixels sont compris entre ces deux droites.

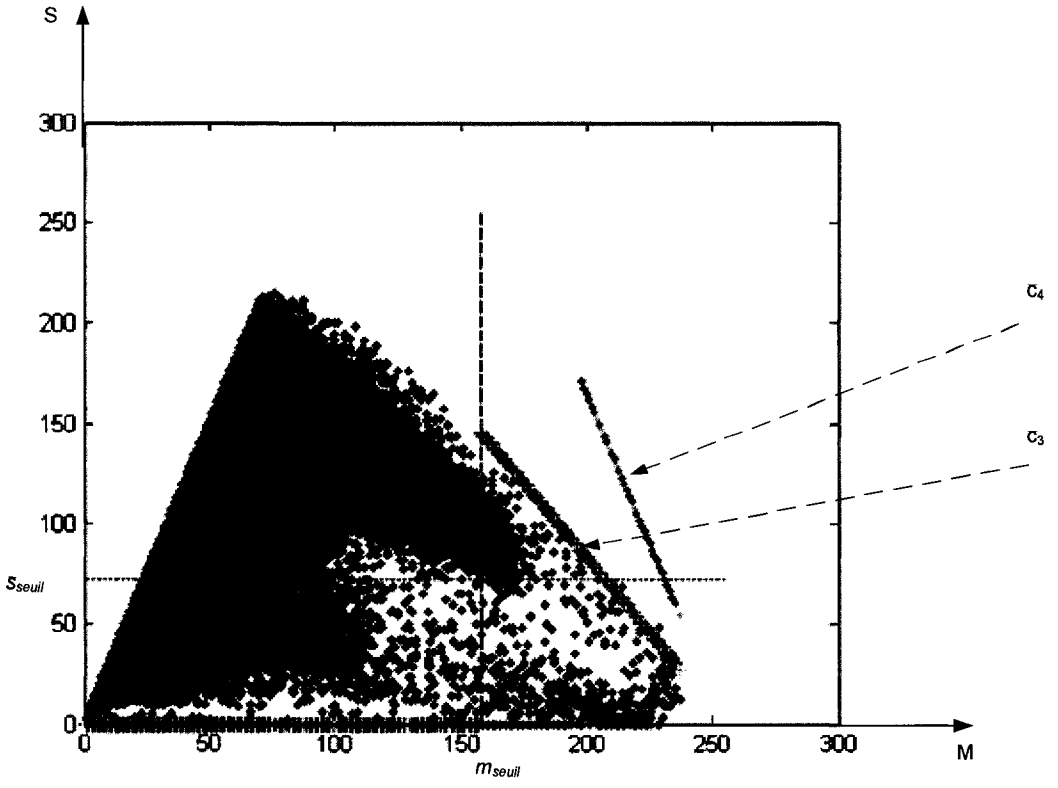


Figure 3-14 Diagramme MS de l'image en figure 2.1

Cela signifie que peu de pixels seront considérés comme spéculaires. Cette remarque est appuyée par la figure 3.15, où le masque spéculaire est peu représentatif des réflexions présentes dans l'image lorsque le seuil défini par l'équation 2.7 est choisie.

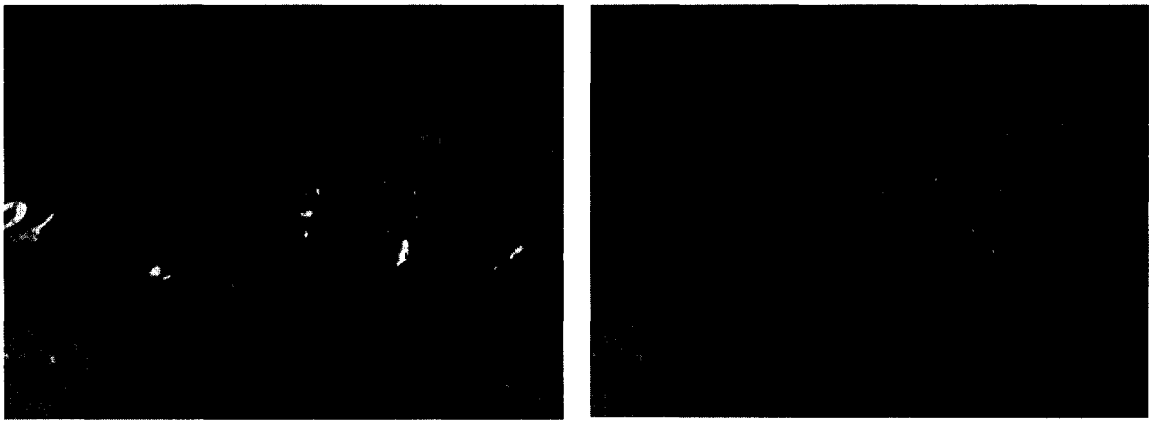


Figure 3-15 Réflexions spéculaires détectées (en bleu) en appliquant les conditions 2.7 (à gauche) et 3.13 (à droite)

L'affichage du masque dans le cas bidimensionnel obéit au même principe que l'affichage du masque dans le cas monodimensionnel. En d'autres termes, les valeurs s_{max} et m_{max} trouvées pour la trame i permettront de créer le masque de la trame i+1. Pour chaque pixel, les valeurs dans le plan M m_p et dans le plan S s_p seront calculées grâce aux équations 3.11 et 2.5. Les comparaisons de l'équation 3.13 seront ensuite évaluées afin de déterminer si le pixel p fait partie de la région spéculaire.

3.4.3 Élargissement du masque

L'élargissement du masque permet de tenir compte du lobe spéculaire. Nous proposons de définir un élargissement statique afin d'alléger les calculs : à chaque fois qu'un pixel faisant partie du masque sera détecté, tous ses voisins directs seront automatiquement

inclus dans le masque. Le processus suivant est effectué afin d'accélérer les calculs. Pour un élargissement de h pixels, le masque spéculaire est inspecté à l'aide d'une fenêtre coulissante de dimension $H \times H$, H étant déterminé par l'équation 3.14 :

$$H = 2h + 1 \tag{3.14}$$

Chaque fois qu'un pixel spéculaire est situé au centre de la fenêtre coulissante, tous les pixels de la fenêtre sont inclus dans le masque spéculaire. La figure 3.16 donne un exemple d'élargissement lorsque $h=1$

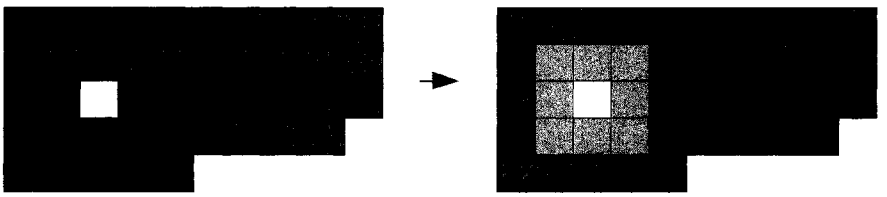


Figure 3-16 Résultat de l'élargissement du masque pour h=1

La partie la plus délicate consiste à trouver la taille idéale de l'élargissement permettant de tenir compte du lobe spéculaire pour toutes les images. La délicatesse vient aussi du fait qu'élargir un masque pourrait causer la fusion involontaire de deux régions voisines. Du point de vue implémentation, l'élargissement utilise le principe de la fenêtre coulissante présentée par Johnston et al afin de garantir un traitement en mode « flux » [21].

Conceptuellement, chaque pixel de la trame de sortie est produit en faisant coulisser une fenêtre de taille NxM par-dessus la trame d'entrée, puis en effectuant une convolution en fonction des pixels de la trame d'entrée situés sous la fenêtre et de l'opérateur de la fenêtre. Le résultat est un pixel assigné au centre de la fenêtre dans la trame de sortie, tel que montré dans la figure 3.17.

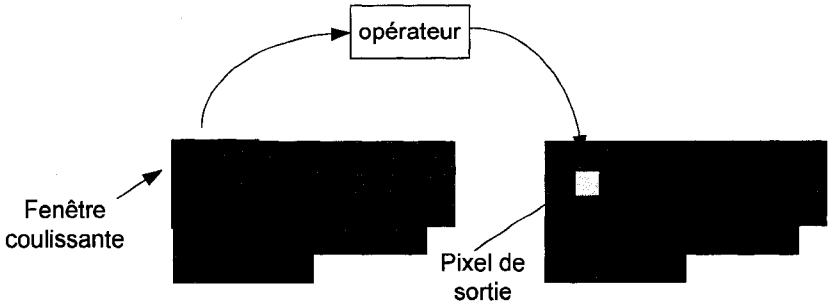


Figure 3-17 Principe de la fenêtre coulissante

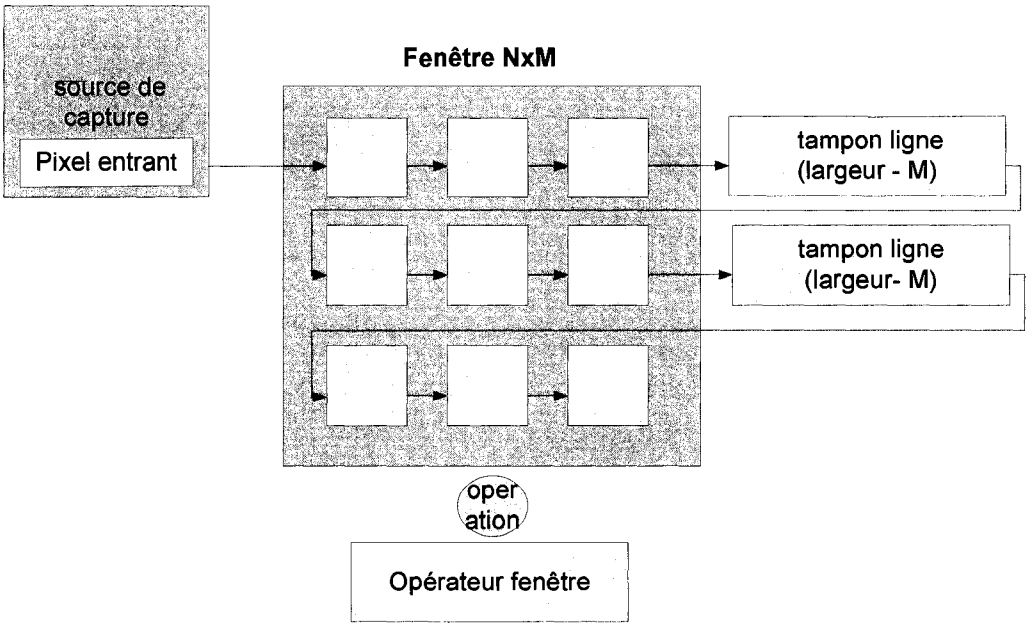


Figure 3-18 Diagramme bloc de la fenêtre coulissante

NxM pixels sont donc nécessaires pour effectuer les calculs chaque fois que la fenêtre est déplacée. Chaque pixel est lu NxM fois. Les contraintes de bande passante rendent l'accès à tous ces pixels en un cycle d'horloge impossible à moins que certaines valeurs soient mises en mémoire. Les pixels d'entrée des N-1 lignes précédentes sont stockés en

utilisant des registres à décalage. Cela permet d'obtenir le diagramme bloc de la figure 3.18.

En utilisant ce principe, le masque spéculaire est inspecté par une fenêtre coulissante de dimension HxH. La taille H nécessaire pour un élargissement de *h* pixel est donnée par l'équation 3.14. L'opération effectuée est un OU logique des pixels contenus dans la fenêtre. Nous avons trouvé qu'une valeur *h=3* était adéquate pour la majorité des images. Cela signifie un délai de traitement de 7 lignes entre le masque d'origine et le masque élargi. La figure 3.19 montre un exemple détaillé d'élargissement en plusieurs itérations lorsque *h=1*. Le pixel blanc représente le pixel spéculaire, les lignes en gras la fenêtre coulissante, et les pixels de couleur grise claire les nouveaux pixels spéculaires après plusieurs itérations successives. Le pixel bleu est le pixel en train d'être évalué.

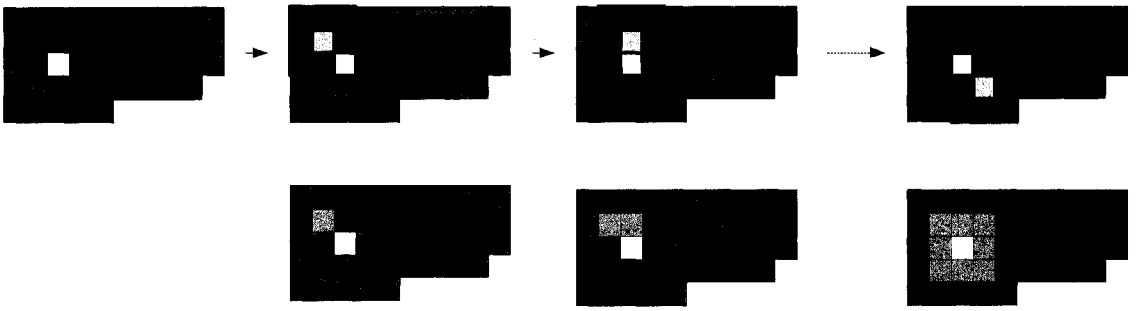


Figure 3-19 Exemple d'élargissement du masque de h=1 pixel

3.5 Correction linéaire

Pour diminuer la quantité de ressources à utiliser au strict minimum, nous effectuons une interpolation linéaire des endroits manquant dans l'image. En d'autres termes, nous opérons ligne par ligne. Il s'agit pour chaque région à corriger dans une ligne de sauvegarder sa largeur *w*, le pixel précédent la réflexion p_b et le pixel suivant la réflexion

p_e(figure 3.20). Ensuite la propagation sera faite de la gauche vers la droite ou de la droite vers la gauche en fonction de la pente a (équation 3.15):

$$a = \frac{p_e - p_b}{w} \tag{3.15}$$

Le pixel p_{i+1} est trouvé à partir du pixel p_i en appliquant la formule 3.16 :

$$p_{i+1} = p_i + a \tag{3.16}$$

Le premier pixel à corriger p_0 obtient la valeur $p_b + a$.

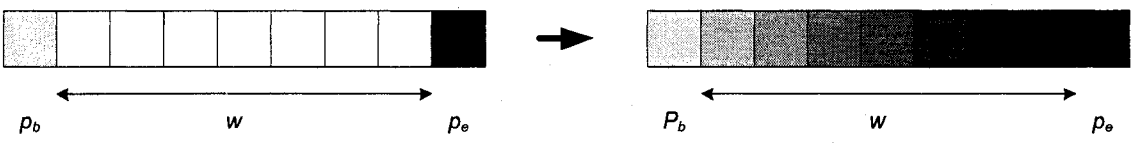


Figure 3-20 Exemple d'interpolation linéaire

Le bloc de correction linéaire d'une trame R, G ou B présenté en figures 3.5 est présenté plus en détail en figure 3.21 ; il passe par deux étapes présentées de manière globale en figure 3.22. La première est le stockage des coefficients p_b, p_e *et* w. Pour ce faire, le masque élargi est lu en parallèle avec la trame d'origine (bloc *calcul coefficients* de la figure 3.21). Pour une trame R,G ou B, deux groupes de RAMs sont disponibles afin de stocker les coefficients des lignes paires et des lignes impaires (bloc *Rams Coefficients* de la figure 3.21). Chaque groupe comporte 3 RAMs de 256x8 bits pour stocker respectivement les coefficients de type p_b, p_e *et* w. Les coefficients sont insérés les uns à la suite des autres chaque fois qu'une région spéculaire est détectée. La ligne en train d'être traitée est insérée, parallèlement au processus de stockage, pixel par pixel dans une FIFO (bloc *Fifos* de la figure 3.21).

La deuxième étape est l'interpolation linéaire(bloc *Correction linéaire* de la figure 3.21). La FIFO est lue en parallèle avec un des deux groupes de RAMs. En effet, lorsque le stockage des coefficients de la ligne i se fait sur le premier groupe, la FIFO (contenant la ligne i-1) est lue parallèlement au deuxième groupe contenant ses coefficients. Lorsque la ligne est parcourue, nous corrigeons une région spéculaire comme suit :

- Une fois le premier pixel spéculaire d'une région spéculaire est détecté, les RAMs sont lues afin de déterminer les coefficients correspondant à la région spéculaire.
- Une fois les valeurs des coefficients reçues, le calcul de la pente a est effectué grâce à l'équation 3.15. Ce calcul est effectué grâce à un module de division qui ne renvoie que la partie entière du résultat. Ce module de division a un délai de deux cycles d'horloge. Pour une division de x par y, l'équation $x := x-y$ est éffectuée jusqu'à ce que x soit inférieur à y. Le nombre de fois que l'opération est effectuée sera la valeur du quotient, et la dernière valeur de x sera la valeur du reste.
- Le premier pixel spéculaire p_0 est remplacé par $p_b + a$. Les autres pixels sont corrigés en utilisant l'équation 3.16.

À la fin du processus nous obtenons un délai d'une ligne (nécessaire au stockage des coefficients) et 6 cycles d'horloges (nécessaire à l'interpolation) entre le masque élargi et la trame corrigée.

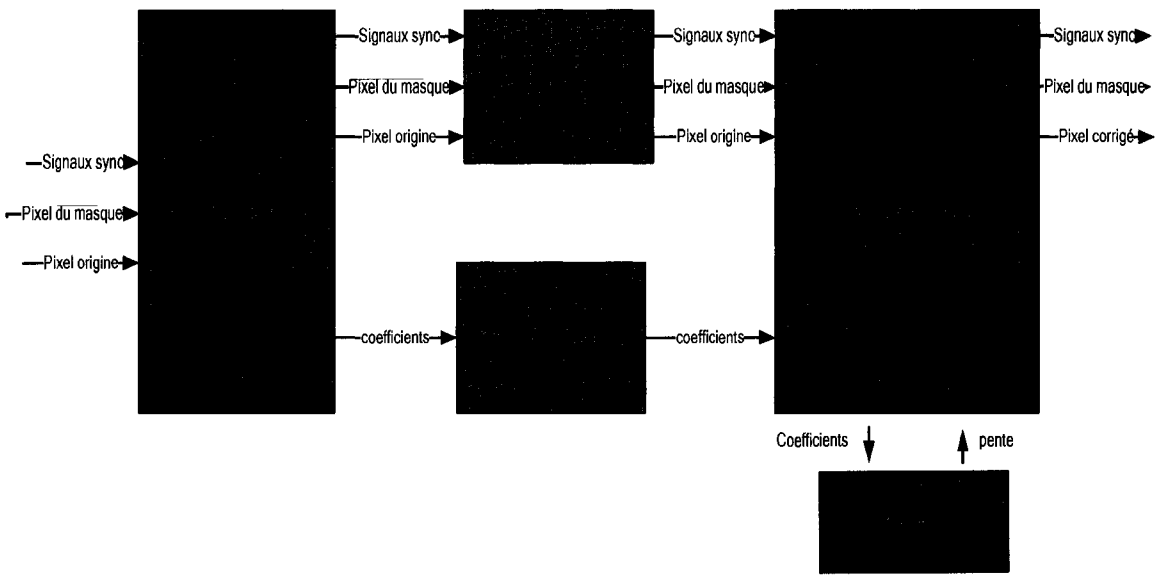


Figure 3-21 Schéma bloc de la correction linéaire d'une trame R,G ou B

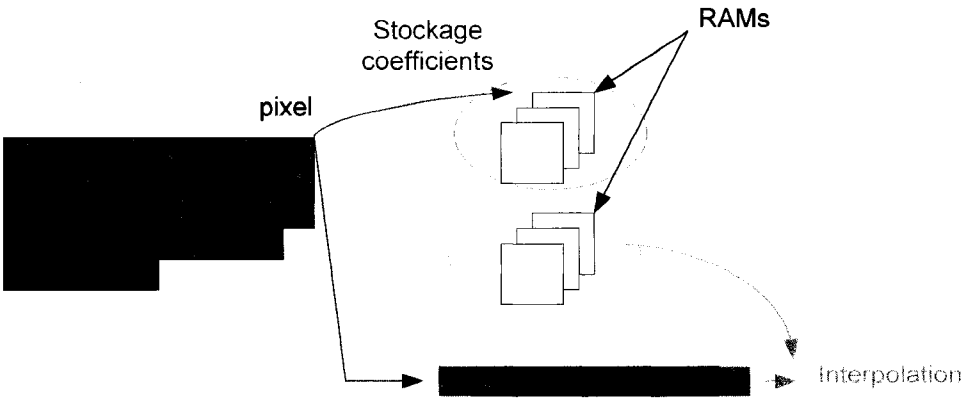


Figure 3-22 Étapes de la correction linéaire

La figure 3.23 présente un diagramme temporel de l'interpolation, après stockage des coefficients :

- Lorsque clk=2, le premier pixel spéculaire p_0 est reçu.

- Lorsque clk=3, la première valeur de la ram est demandée grâce à adr_1.
- Lorsque clk=4, les coefficients stockés à adr_1 sont reçus.
- Lorsque clk=5, le diviseur et le quotient sont transmis au bloc de division
- Lorsque clk=7, le résultat de la division a est reçu.
- À partir de clk=8, tous les pixels spéculaires sont corrigés grâce à a. La correction s'arrête lorsqu'on détecte la fin de la région spéculaire.

Le processus de correction linéaire est répliqué 3 fois afin de traiter en parallèle les trois trames RGB.

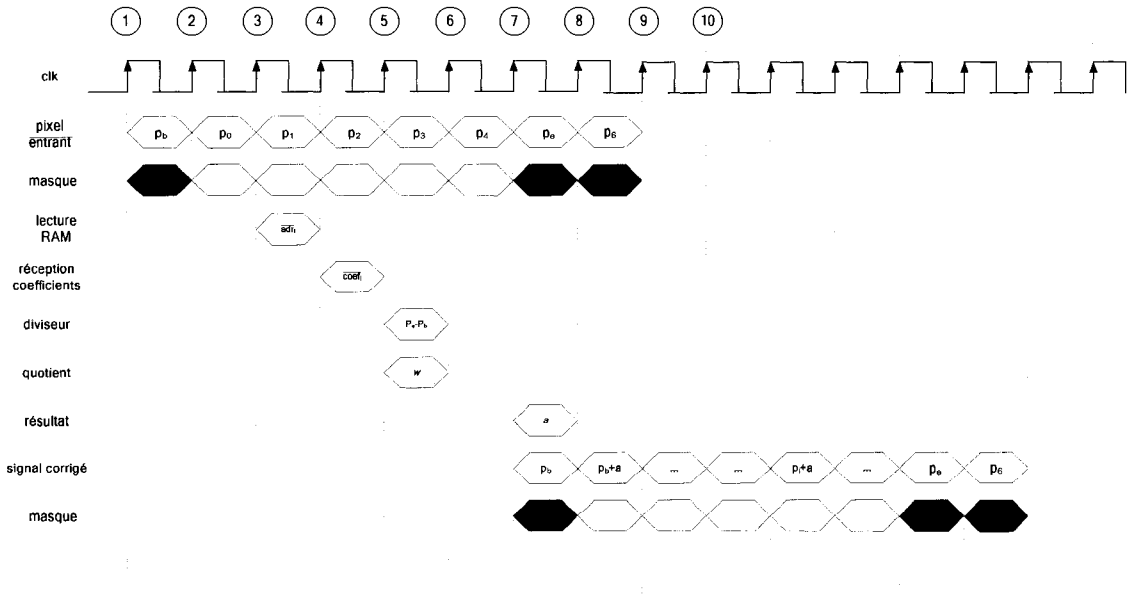


Figure 3-23 Diagramme temporel d'une interpolation linéaire

3.6 Lissage

La correction n'intervenant que dans la dimension horizontale, il est nécessaire d'y ajouter une composante verticale. Cela s'effectue en remplaçant la valeur actuelle de

chaque pixel spéculaire par la moyenne de ses voisins à l'aide d'une fenêtre coulissante de taille 3x3. Il faut donc stocker 2 lignes pour effectuer la moyenne des pixels en mode « flux ». Cette procédure de lissage est effectuée un nombre limité de fois pour éviter une grande consommation en ressources. Nous l'effectuons 5 fois, ce qui équivaut à un délai de 15 lignes.

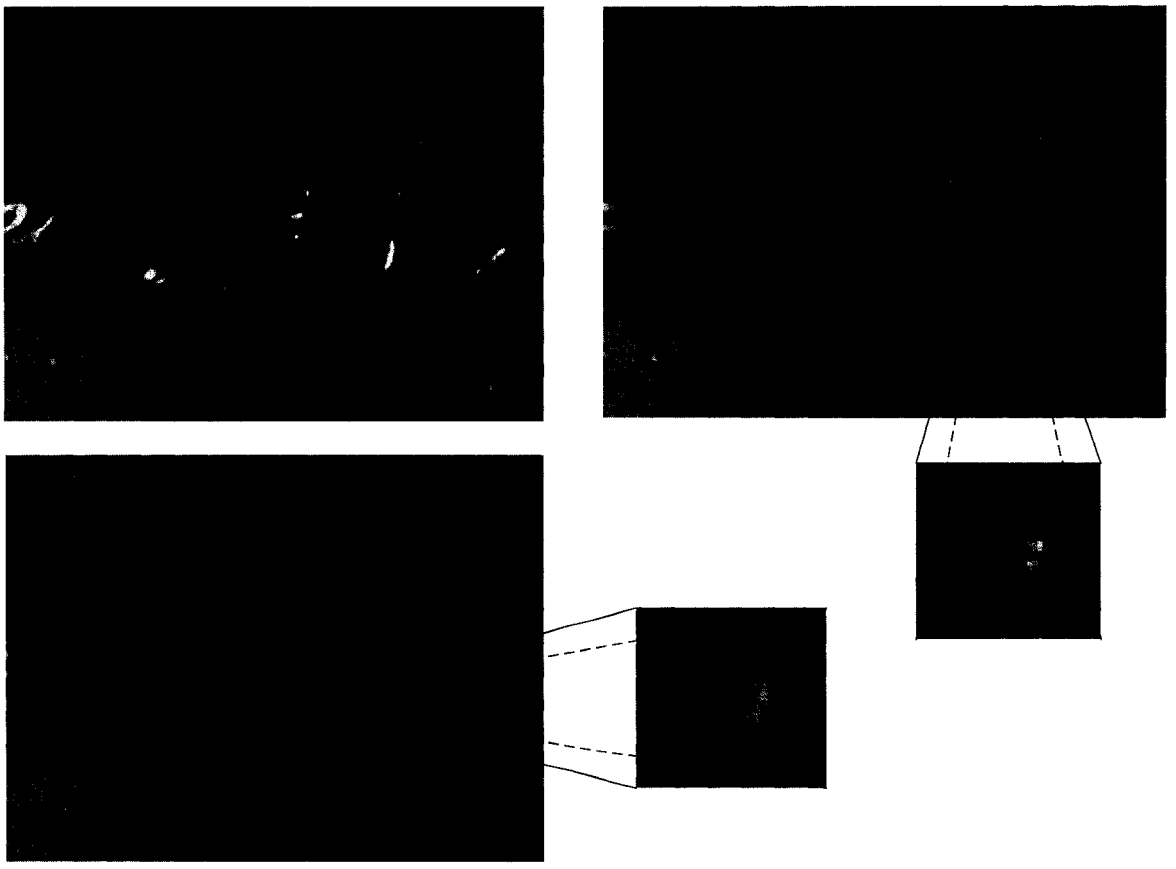


Figure 3-24 Correction par interpolation linéaire (en haut à droite), puis lissage (en bas) de l'image en haut à gauche

La figure 3.24 montre les résultats d'une correction après interpolation linéaire, puis après lissage. Ici, on part de réflexions spéculaires précorrigées contrairement à la méthode définie dans [14]. Cela assouplit les conditions d'arrêt de la correction. On peut

remarquer sur cette figure que les corrections effectuées par l'interpolation linéaire donnent des lignes, lignes atténuées et rendues homogènes au reste de l'image après le lissage.

3.7 Conclusion

Les algorithmes choisis pour implémenter notre système ont été optimisés dans ce chapitre, optimisation qui conduit à une implémentation matérielle qui limite l'utilisation des ressources. Le chapitre suivant permet de vérifier que les résultats obtenus correspondent à nos attentes.

Chapitre 4 Résultats et Discussion

Ce chapitre présente les résultats obtenus après implémentation des algorithmes de détection et de correction. Après avoir donné un apercu de l'environnement de vérification, nous effectuons la vérification des systèmes de détection et du système de correction. Nous présentons ensuite les performances matérielles des différents systèmes suivi d'une comparaison des deux systèmes de détection. Cette comparaison conduit à un choix d'implémentation final.

4.1 Environnement de vérification

La figure 4.1 permet d'avoir un aperçu des étapes nécessaire à l'implémentation et la vérification du système.

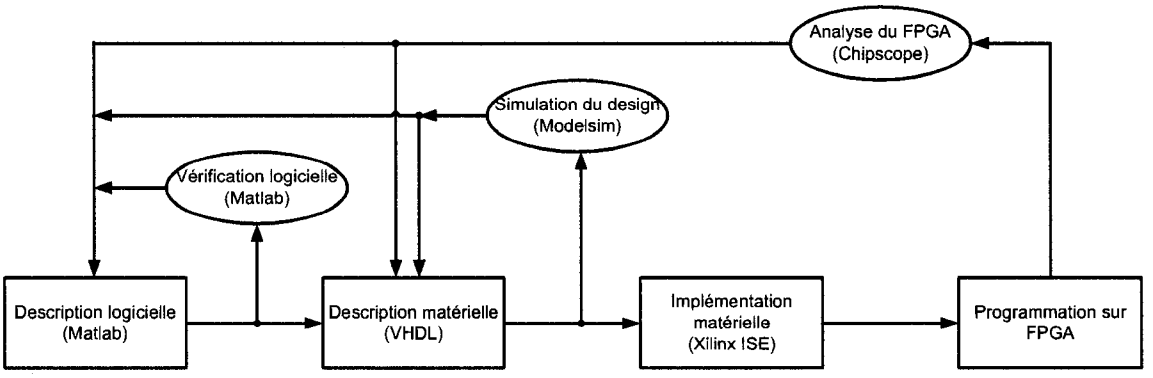


Figure 4-1 Flot de conception

Les différents algorithmes sont premièrement testés de manière logicielle avant leur implémentation matérielle. Cela permet de profiter de la flexibilité logicielle afin de diminuer la phase de vérification matérielle. Nous utilisons Matlab pour la vérification

logicielle car c'est le candidat idéal pour l'implémentation et la simulation rapide d'algorithmes de traitement d'image. Afin de simuler le plus les conditions matérielles, nous utilisons uniquement des données entières définies sur une plage de bits déterminée, généralement 8 bits. Nous utilisons aussi le moins possible les fonctions pré-existantes dans Matlab afin de faciliter l'importation des algorithmes dans l'outil de synthèse matériel. Nous évitons aussi l'utilisation de méthode non implémentables directement telles que des méthodes récursives.

Après avoir validé l'implémentation logicielle, le système est décrit de manière matérielle grâce au langage VHDL, synthetisé avec Xilinx ISE, et simulé grâce à Modelsim. Pour ce faire, nous simulons chaque boc indépendamment et comparons son résultat à son équivalent sur Matlab. Une fois la procédure de vérification par bloc approuvée, nous simulons le système dans son ensemble. Pour cela, nous inscrivons premièrement une image RGB dans 3 fichiers, un fichier par trame de couleur. Ensuite, nous créons un banc de test qui lit à une fréquence fixe le contenu des fichiers et transmet les données RGB et les signaux de synchronisation au système testé. L'image est parcourue deux fois, une première fois pour la détection, une seconde fois pour la correction. Les pixels sortant du bloc de correction sont inscrits dans un fichier texte puis retransformés en une image RGB grâce à Matlab.

Il est ensuite possible de vérifier le système une fois implémenté sur FPGA. Cela se fait grâce à l'outil de Xilinx Chipscope. Il s'agit d'un outil qui insère un analyseur logique et un analyseur de bus directement dans le design afin de capturer les signaux désirés après la programmation du FPGA. Les signaux sont capturés après le déclenchement d'un

événement spécifié (front montant d'un signal donné par exemple), ou manuellement, sur commande de l'utilisateur. Les signaux capturés peuvent ensuite être analysés grâce à l'analyseur logique.

4.2 Vérification du système de détection

4.2.1 Détection monodimensionelle

4.2.1.1 Exemples bloc par bloc

Afin de valider nos résultats nous prenons premièrement une image aléatoire présentée en figure 4.2 sur laquelle nous appliquons les différentes fonctions de notre système. Il s'agit d'une image 10x10 générée grâce à Matlab.

Figure 4-2 Image aléatoire servant pour les tests

Figure 4-3 Résultats de la transformée en niveaux de gris sur matlab (à gauche) et sur modelsim (à droite)

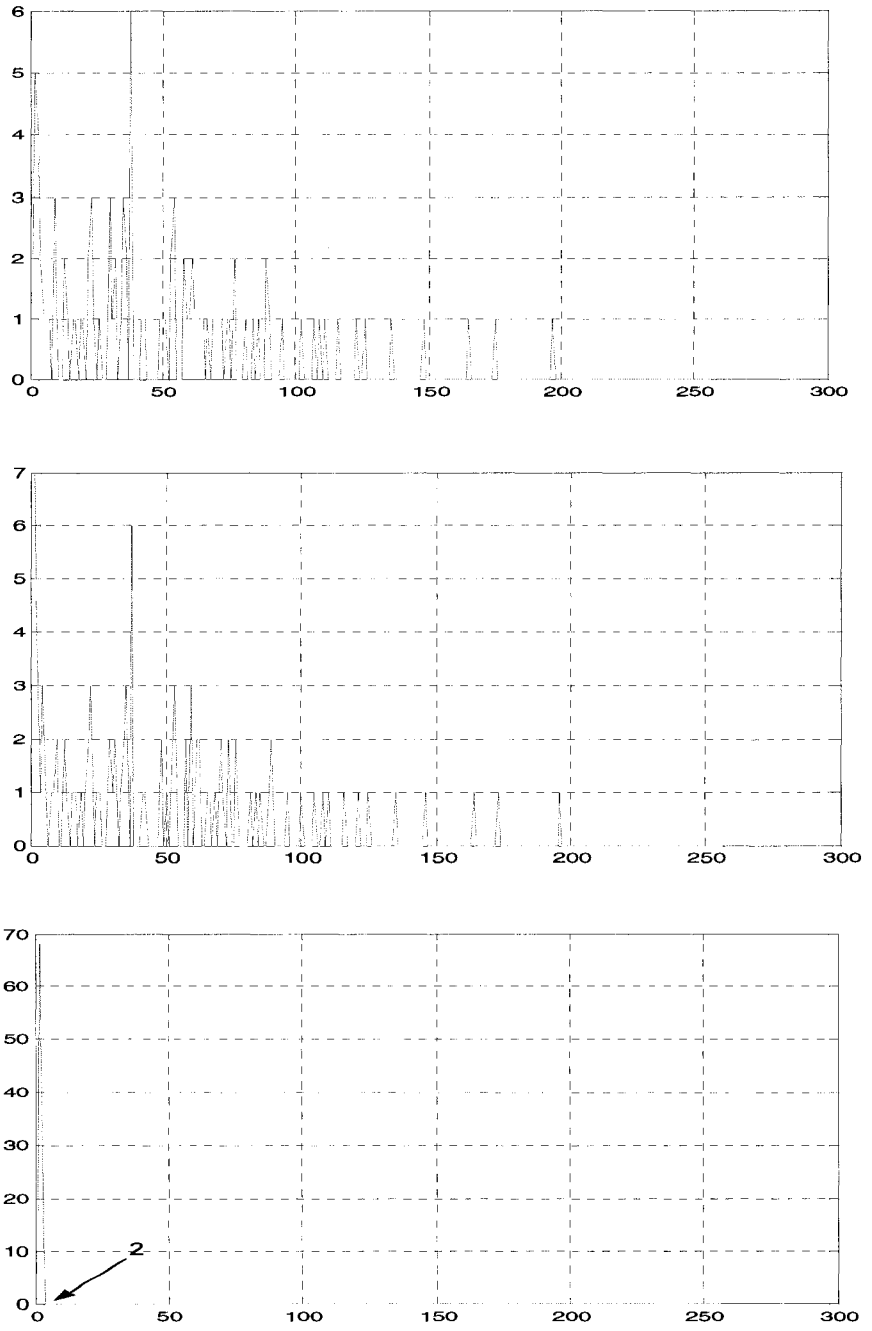


Figure 4-4 Histogrammes résultant de la transformation en niveau de gris ; en haut: issu de matlab ; au milieu: issu de modelsim ; en bas: issu de la différence entre la transformation en niveau de gris et de modelsim et matlab

Nous pouvons donc confirmer en comparant les résultats sur matlab à ceux obtenus grâce à la simulation modelsim que notre système présente une erreur de précision lors du passage du domaine RGB au domaine Y, en effectuant un rehaussement préalable. La figure 4.3 permet de comparer l'image en niveau de gris générée par Matlab et celle obtenue grâce à Modelsim.

On obtient une erreur maximale de 2 entre les deux images ; cette erreur est une erreur cumulée du rehaussement, puis de la transformation en niveaux de gris. Il n'est malheureusement pas possible d'observer cette différence visuellement, celle-ci étant minime.

Nous déduisons des histogrammes des transformées en niveaux de gris obtenus. Les deux histogrammes théorique et réel présentés en figure 4.4 sont similaires. Afin de mieux comparer les deux transformations en niveaux de gris, nous présentons un histogramme issu de la différence en valeur absolue entre les deux images. Celui-ci montre que l'erreur maximale est de 2.

Nous effectuons ensuite une dérivation des histogrammes obtenus sans débruitage préalable. Malgré l'erreur engendrée par la transformation en histogramme, les résultats théoriques et réels de la dérivée de celui-ci sont similaires.

Le bloc de débruitage appliqué sur un histogramme généré aléatoirement donne des résultats similaires à ceux obtenus théoriquement.

Nous pouvons donc conclure que les blocs testés indépendamment fonctionnement comme prévu.

4.2.1.2 Exemples sur des images réelles

Bien que fonctionnant correctement lorsqu'ils sont simulés indépendamment, les résultats obtenus lorsqu'une image passe par l'algorithme de détection au complet présentent des différences avec les résultats théoriques. En observant les histogrammes obtenus à partir des images présentées en figure 4.5, on se rend compte que l'allure de la courbe, et par la même occasion la répartition générale des pixels, est conservée (figures 4.6 et 4.7).

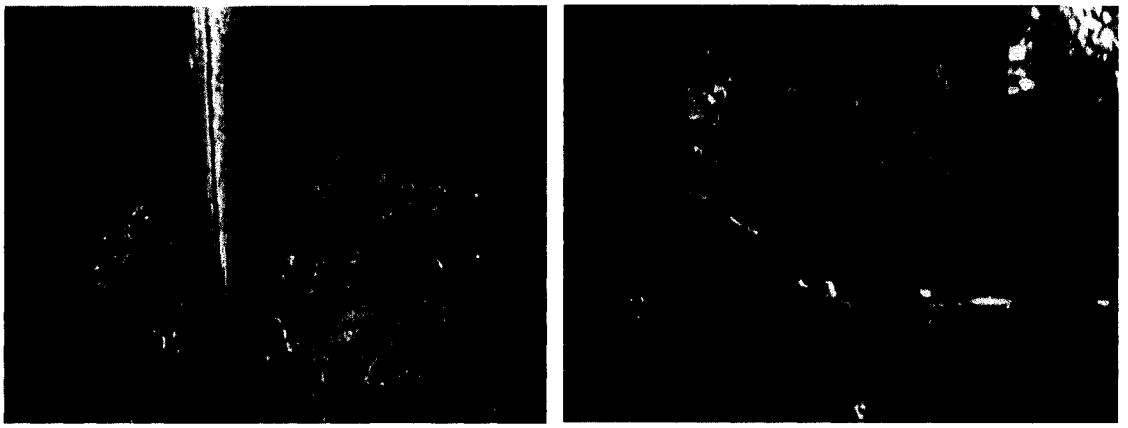


Figure 4-5 images utilisés pour tester la détection monodimensionnelle

Cependant, les quantités de pixel par plage d'intensité varient. Cette variation est due à l'erreur faite lors du rehaussement et de la transformation du pixel en niveau de gris. En effet, un pixel qui devrait normalement se retrouver dans une zone d'intensité se retrouvera facilement dans une autre, voisine. Cette légère variation est atténuée par le débruitage qui ne conserve que l'allure générale de la courbe, ce qui devrait permettre d'obtenir le même début de bosse spéculaire (figure 4.6).

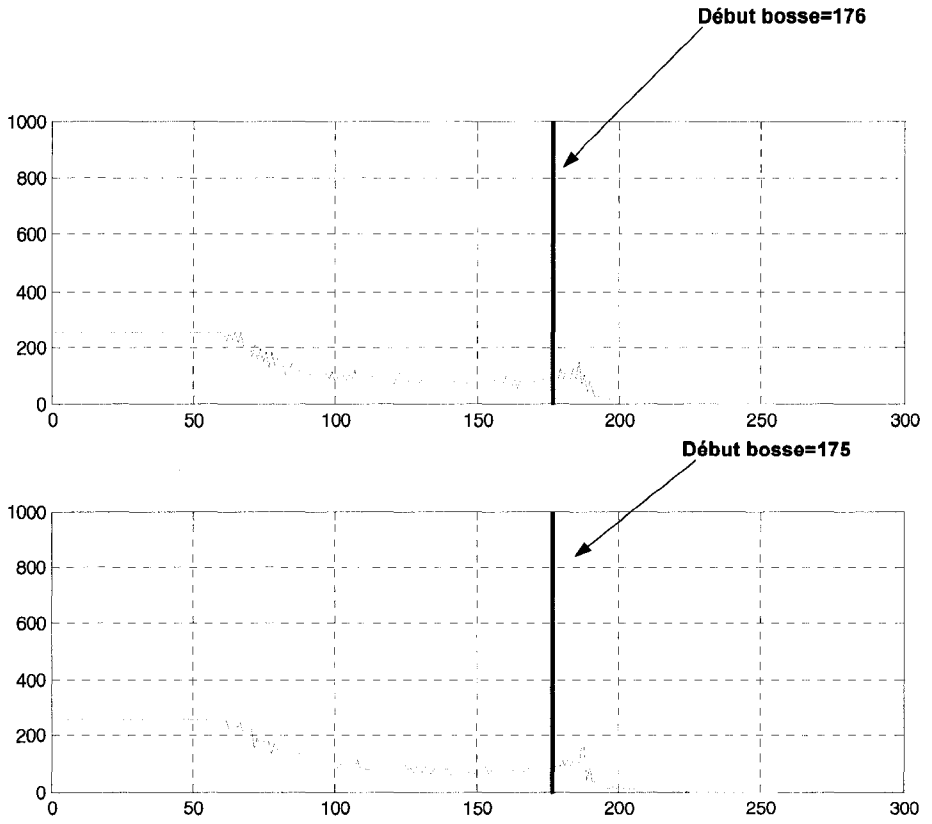


Figure 4-6 Histogrammes obtenus à partir de Modelsim (en haut) et Matlab (en bas) pour l'image de droite de la figure 4.5

Malheureusement, Il existe certaines images pour lesquelles il n'y a pas une bonne répartition des pixels dans l'histogramme. En d'autres termes, on n'observe pas toujours une zone où sont localisés les pixels de la région diffuse bien démarquée de la zone spéculaire. Cela entraine une mauvaise détection du début de la région spéculaire comme dans le cas de l'image de gauche de la figure 4.5 dont les histogrammes sont présentés en figure 4.7.

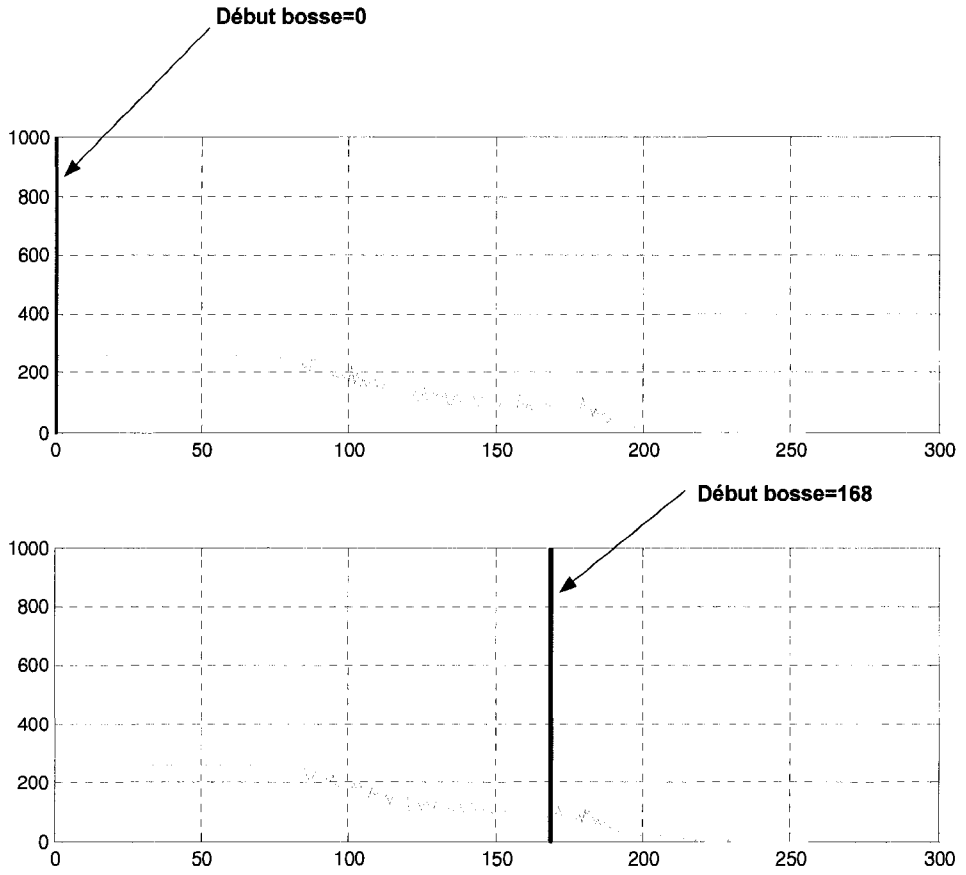


Figure 4-7 Histogrammes obtenus à partir de Modelsim (en haut) et Matlab (en bas) pour l'image de gauche de la figure 4.5

En observant les résultats obtenus par Chipscope (figure 4.8), on se rend compte que l'histogramme varie, pour la même image. Cette variation est observée à l'entrée du système. En effet on peut constater que les pixels envoyés par le décodeur vidéo varient légèrement d'une trame à l'autre, pour la même image affichée à l'écran. Cette variation est due aux différentes conversions analogiques numériques qui surviennent dans le lecteur DVD, dans le décodeur vidéo et le passage YCrCB à RGB. Cela entraîne pour la

même image différents débuts de région spéculaire. Bien qu'il détecte les régions spéculaires, le masque obtenu est donc souvent instable.

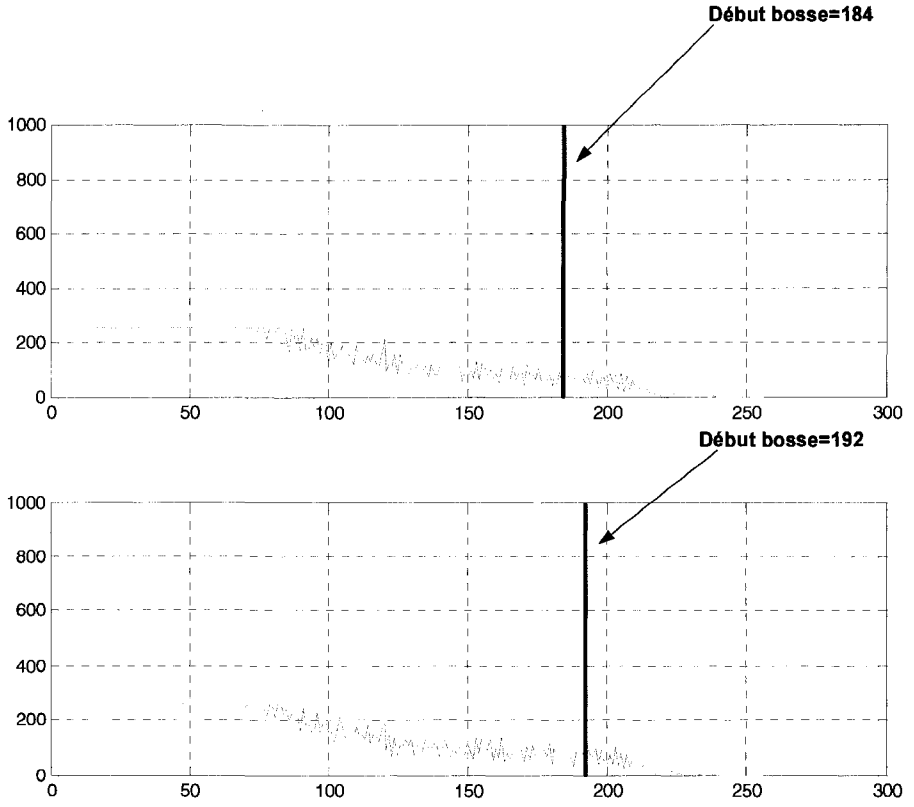


Figure 4-8 Histogrammes de l'image de gauche de la figure 4.5 obtenus à partir de Chiscope. Les débuts de bosse spéculaires sont calculés sur matlab

4.2.2 Détection bidimensionnelle

Les seuls calculs effectués lors de la détection bidimensionnelle sont les calculs du plan M et du plan S afin de déterminer leurs valeurs maximales respectives. Les valeurs maximales de S et M sont similaires, que ce soit dans matlab ou modelsim. En effet on observe les données présentées dans le tableau 4.1.

Les erreurs de précision dues à la transformation en M et en S des pixels de l'image en figure 4.9 sont telles que prévues dans le chapitre 3. Nous obtenons donc des plans S et M visuellement identiques à leur résultat théorique comme on peut le constater en observant les figures 4.10 et 4.11

Figure 4-9 Image utilisée pour les tests du plan S et du plan M

Tableau 4-1 Valeurs maximale des plans S et M de l'image en figure 4.8

	modelsim	matlab
smax	214	215
mmax	238	238

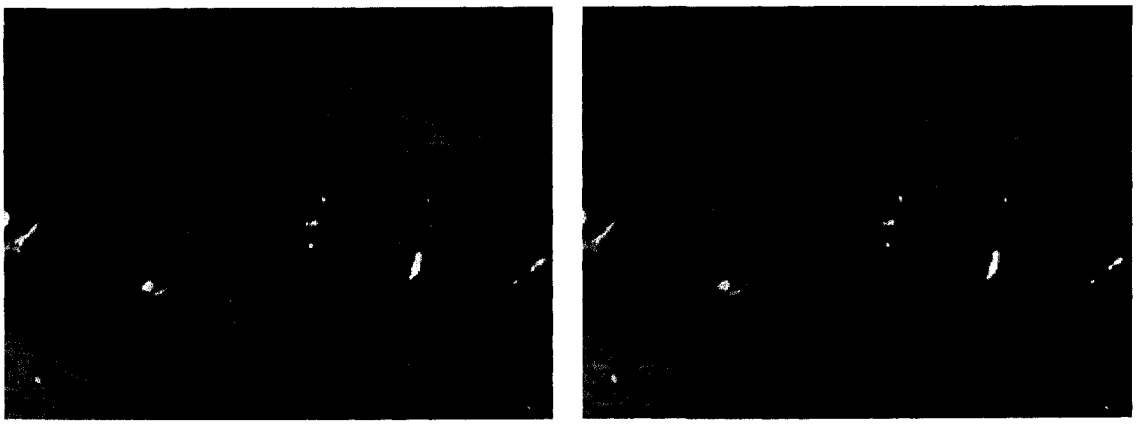


Figure 4-10 Implémentation matlab(à gauche) et modelsim (à droite) du plan M de l'image en figure 4.9

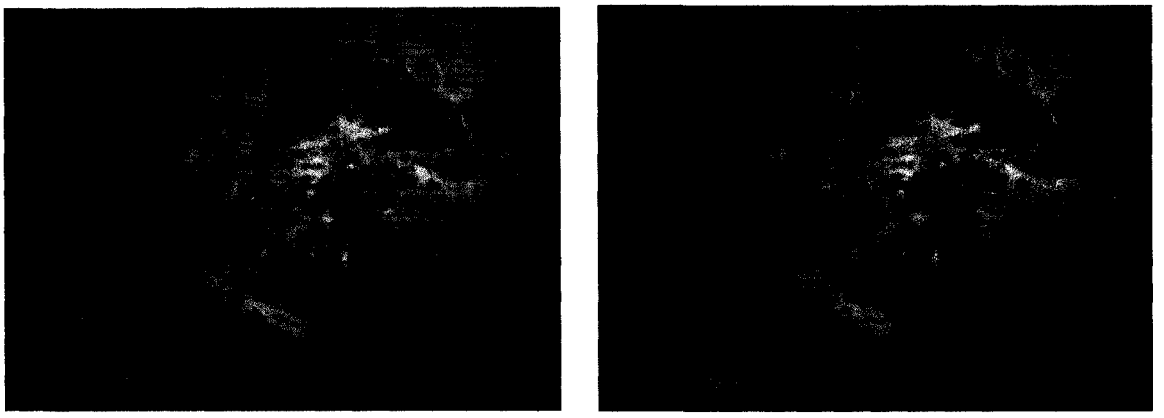


Figure 4-11 Implémentation matlab(à gauche) et modelsim (à droite) du plan S l'image en figure 4.9

Comme attendu, les masques sur modelsim et matlab sont légèrement différents. Les deux réussissent cependant à capturer les principales régions spéculaires (figure 4.12).

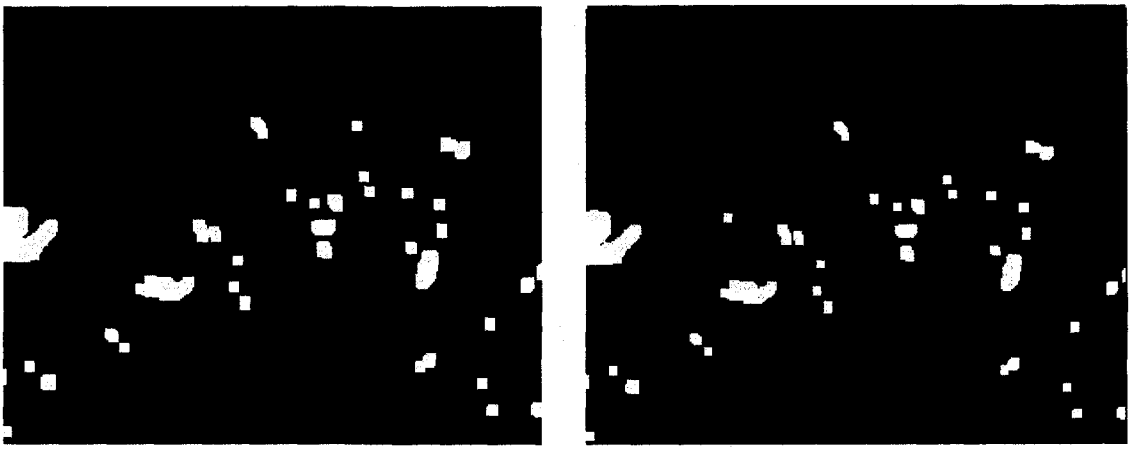


Figure 4-12 Implémentation matlab(à gauche) et modelsim (à droite) du masque spéculaire de l'image en figure 4.9

Comme énoncé précédemment lors de la simulation de la détection monodimensionnelle, on constate que le masque résultant scintille. Cependant dans ce cas, les scintillements sont moins fréquents car ici la détection de la région spéculaire ne tient pas compte de toutes les variations possibles dans l'image. Cet algorithme est donc plus robuste face aux bruits sur des pixels isolés.

4.3 Vérification du système de correction

L'algorithme de correction présente des résultats très satisfaisants dans l'ensemble. Les régions détectées sont corrigées linéairement puis lissées. Il est clair que des étapes supplémentaires de lissage permettraient d'améliorer les résultats obtenus.

Par exemple, Il est difficile pour l'algorithme de corriger efficacement des zones très larges. Cela peut s'observer sur la figure 4.13 où même après la procédure de lissage,

l'aspect ligne de la correction est toujours présent. On distingue donc des variations de couleurs brusques dans le sens vertical.

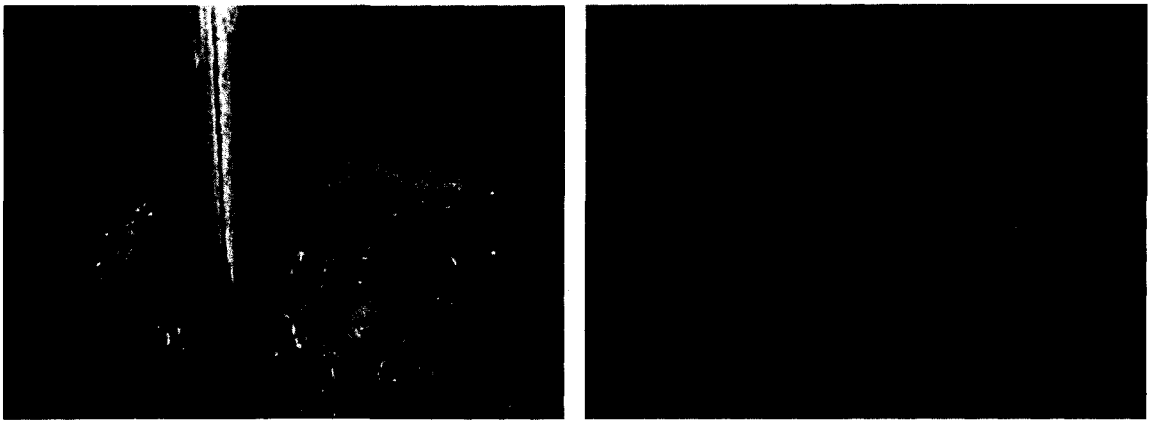


Figure 4-13 Manque d'efficacité du lissage pour de larges régions spéculaires; à gauche: image d'origine; à droite: image corrigée après implémentation matérielle

Augmenter les étapes de lissage permettrait d'annuler complètement l'effet ligne encore observable. La région corrigée présenterait la même texture que la région l'entourant. La figure 4.14 présente une correction effectuée avec 100 étapes de lissage sur Matlab. Il est possible d'observer que l'effet ligne a disparu et que la texture de l'objet métallique est plus uniforme. Compte tenu des resssources disponible il est malheureusement impossible d'effectuer cette correction sur FPGA.

Figure 4-14 Correction améliorée avec des étapes de lissage supplémentaire; à gauche: image d'origine; à droite: image corrigée avec 100 étapes de lissage

L'algorithme de correction fonctionne bien lorsque la région spéculaire est située à l'intérieur d'un objet. Lorsqu'elle est située à l'intersection de deux objets différents, l'information provenant d'un objet peuvent être propagées dans l'autre. Ce phénomène peut exister naturellement, ou encore être provoqué lors de l'élargissement du masque qui ne tient pas compte des bordures des objets. Le masque pourra donc des fois être prolongé dans l'objet voisin.

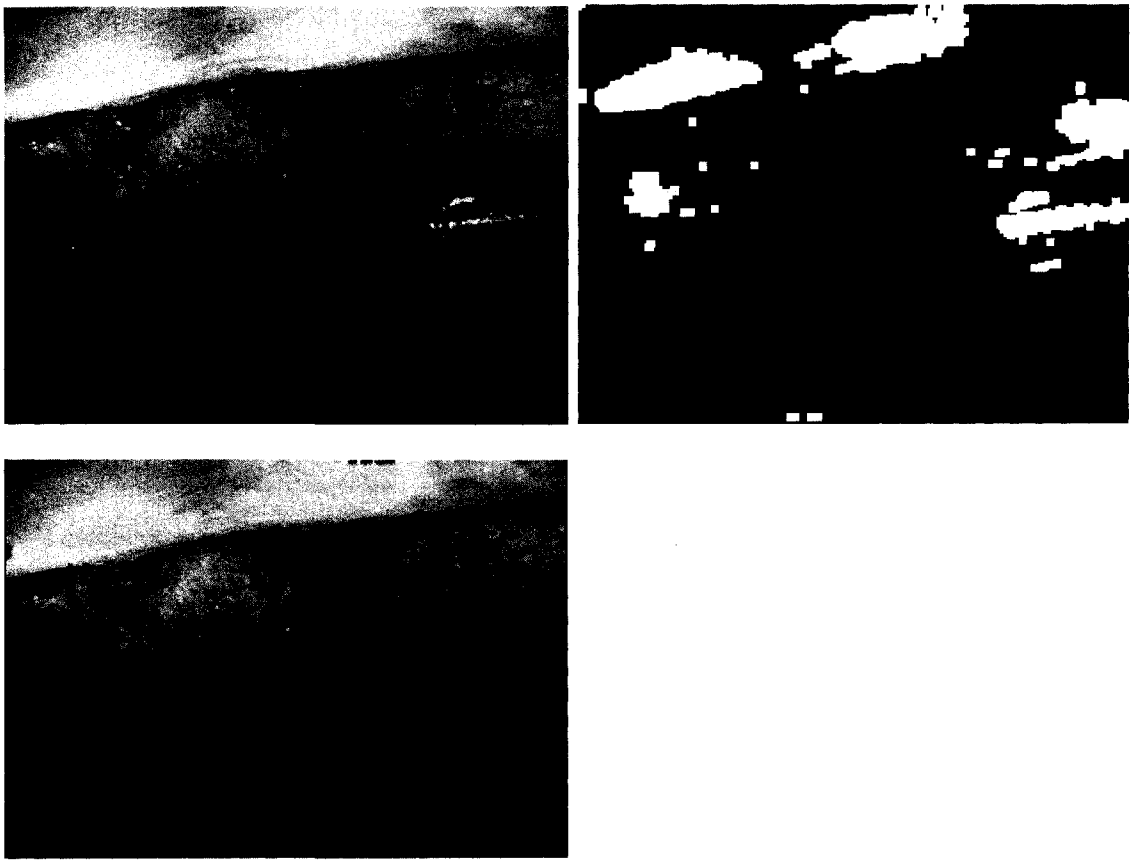


Figure 4-15 Correction de la réflexion spéculaire par la mauvaise couleur : la réflexion présente sur l'outil métallique est corrigé par l'information contenu dans la zone rouge à gauche de l'objet

La figure 4.15 montre un exemple de ce défaut. On constate que la réflexion spéculaire

présente sur l'objet métallique est corrigée par la couleur rouge à cause du masque qui s'élargit vers la gauche et englobe la zone rouge.

On observe aussi sur l'écran vidéo un scintillement prévisible des zones corrigées. Elles sont en effet dues à l'imperfection du masque spéculaire utilisé pour la détection.

4.4 Performances matérielles

Pour évaluer les performances matérielles nous effectuons le processus de synthèse des algorithmes de détection monodimensionnelle, détection bidimensionnelle, puis de détection bidimensionnelle suivi de la correction. Nous obtenons le tableau 4.2 qui présente les ressources matérielles utilisées obtenues après la phase de synthèse de l'implémentation matérielle [25]. Les ressources utilisées par l'algorithme de correction sont obtenues en soustrayant les ressources utilisées pour la détection bidimensionnelle à celles utilisées pour la détection bidimensionnelle suivie de la correction.

Afin d'obtenir des ressources indépendante de la structure du FPGA, les ressources présentées ne tiennent pas compte des ressources nécessaires au routage. Il s'agit uniquement de la logique du circuit. Pour la même raison nous préférons nous baser sur le nombre de flip-flops et de LUTs utilisées plutôt que sur le nombre de slices.

On peut constater que l'algorithme de détection monodimensionnelle utilise 10 fois plus de ressources que l'algorithme de détection bidimensionnelle. Cela est dû à la phase de traitement de l'histogramme qui s'effectue lorsque le système entre en mode « vidéo inactive ».

Tableau 4-2 Ressources utilisées pour chaque implémentation matérielle sur un FPGA Xilinx Virtex 2 pro XC2VP30

algorithme	Flip-flops (27,392 disponibles)	LUTs (27,392 disponibles)	Slices (13696 disponibles)	Brams (2,448 Kb disponibles)
Détection monodimensionnelle	6,002 (21%)	10,531 (38%)	6,844 (49%)	162Kb (6%)
Detection bidimensionnelle	571 (2%)	1,032 (3%)	586 (4%)	108Kb (4%)
Correction	3,641 (13%)	24,044 (88%)	12,775 (93%)	1,018Kb (42%)
Detection bidimensionnelle et correction	4,212 (15%)	25,076 (91%)	13,361 (97%)	1,126Kb (46%)

Le délai de traitement du système est provoqué par la phase de correction. En additionnant les différents délais établis dans le chapitre 3, nous obtenons un délai de 25 lignes de 858 pixels (avec 720 pixels actifs). En termes de temps, nous avons donc un délai de 0.8ms entre la trame d'entrée non corrigée et la trame de sortie corrigée. Après optimisation du placement et routage, notre système utilise 91% des LUTs disponibles et a une fréquence maximale d'opération de 32 MHz, ce qui excède le minimum requis de 27 MHz pour des opérations en temps réel.

4.5 Algorithme de traitement choisi

L'algorithme de détection monodimensionnelle est celui devant présenter les résultats les plus précis grâce à sa procédure de traitement de l'histogramme. En effet le débruitage de l'histogramme suivi de sa segmentation permet théoriquement d'obtenir

la position exacte des régions spéculaires. Malheureusement, cela est uniquement possible si plusieurs conditions sont respectées :

- La région de l'histogramme où se situent les pixels spéculaires doit être bien délimitée de la région diffuse.
- Le débruitage doit être précis afin de ne pas éliminer accidentellement les régions importantes de l'histogramme
- Les valeurs nécessaires à la création de l'histogramme doivent être précises (utilisation si possible de données décimales, bonne conversion analogie-numérique à l'entrée du système)

Ces conditions ne sont pas facilement respectables lorsqu'on veut limiter l'utilisation des ressources du FPGA. De plus, nous devons aussi tenir compte des ressources nécessaires à l'implémentation de l'algorithme de correction. En effet, nous constatons que les ressources nécessaires à l'implémentation d'un système de détection monodimensionnelle, tel qu'il est conçu actuellement, suivi d'un algorithme de correction sont supérieures aux ressources disponibles dans le FPGA. Nous nous tournons donc vers l'algorithme bidimensionnel qui offre une meilleure robustesse tout en utilisant moins de ressources. Cela permet de faire tenir tout notre système à l'intérieur du FPGA tout en donnant des résultats en temps réel.

Conclusion

Ce mémoire a proposé un algorithme permettant de détecter puis de corriger des zones de réflexions spéculaires présentes dans des séquences d'images endoscopiques. Le défi est d'implémenter cet algorithme sur un FPGA de petite taille et de le faire fonctionner en temps réel afin qu'il puisse servir dans une salle d'opération, particulièrement dans la constitution d'un environnement à réalité augmentée.

Deux méthodes de détection ont été proposées. La méthode à histogramme monodimensionnelle se base sur le fait que le pic spéculaire est observable dans les histogrammes RGB d'une trame. Il suffit alors de détecter la zone de début de ce pic grâce à un seuillage des trois histogrammes, puis faire une descente en intensité afin de tenir compte du lobe spéculaire. Cette méthode a déjà été testée de manière logicielle. Quelques modifications ont été effectuées afin de réduire la consommation en mémoire et de la rendre plus rapide. Il s'agit de l'utilisation d'une seule trame de couleur au lieu de trois, et de la modification de l'algorithme de descente en intensité. Une particularité de cette méthode est l'utilisation de la zone de vidéo inactive afin d'effectuer les calculs les plus complexes. La deuxième méthode utilisée pour la détection est la méthode bidimensionnelle. Elle consiste à utiliser conjointement le plan saturation et intensité (nuances de gris) d'une trame afin de détecter les zones de réflexions spéculaires. En effet un seuillage à la fois sur la trame de saturation et la trame en nuance de gris permet d'isoler la zone de réflexions spéculaires. Pour l'adapter à une implémentation matérielle et au type d'images sur lesquelles nous travaillons, l'algorithme permettant le

calcul des seuils est modifié. Ici nous n'attendons pas d'obtenir entièrement de calculer un plan avant de calculer l'autre. Les deux méthodes de détection fonctionnent en temps réel.

Une méthode de correction a été proposée. Celle-ci se base sur le principe de restauration d'image déjà utilisé dans d'autres algorithmes qui consiste à utiliser l'information provenant du contour d'une zone spéculaire et prolonger cette information à l'intérieur de la zone, ceci dans les deux directions verticale et horizontale. Afin de tenir compte des ressources limitées, la méthode implémentée ne travaille pas simultanément dans les deux directions. Elle effectue premièrment une interpolation linéaire dans la direction horizontale. Ensuite un lissage permet de tenir compte de la direction verticale. Cette méthode fonctionne en temps réel.

La méthode monodimensionnelle étant gourmande en ressources, le système implémenté est constitué de la méthode de détection bidimensionnelle suivie de la correction. La particularité de ce système est que les deux algorithmes fonctionnent en parallèle, sur la même trame. En effet, les informations provenant de l'algorithme de détection appliquée sur la trame i permettent d'appliquer la correction sur la trame i+1. Le système au complet utilise 91% des ressources du FPGA.

Bien que le système fonctionne, plusieurs points peuvent être améliorés. Premièrement, l'amélioration du processus de capture vidéo. En effet on observe un effet de scintillement lors de la création du masque. Ensuite, l'utilisation d'une méthode de descente en intensité plus efficace permettrait d'obtenir un meilleur masque spéculaire. Cela permettrait d'éviter que lors de la correction, l'information provenant d'une zone

soit propagée dans la zone voisine. Concernant la correction, on pourrait tester l'implémentation d'un algorithme de correction encore plus efficace, telle que la méthode Navier-Stokes. Cela serait cependant difficile à obtenir avec le présent FPGA, très limité en terme de ressources.

Références

[1] Société Française de Chirurgie Endoscopique, "Qu'est ce que la coeliochirurgie? " [En ligne]. Disponible: www.lasfce.com/sfce/coeliochirurgie/

[2] I. Bricault, G. Ferretti, P. Cinquin, , "Registration of real and CT-derived virtual bronchoscopic images to assist transbronchial biopsy," *Medical Imaging, IEEE Transactions on* , vol.17, no.5, pp.703-714, Oct 1998.

[3] J.P. Helferty, A.J. Sherbondy, A.P. Kiraly, J.Z. Turlington, E.A. Hoffman, G. McLennan, W.E. Higgins, "Image-guided endoscopy for lung-cancer assessment," *Image Processing, 2001. Proceedings. 2001 International Conference on* , vol.2, no., pp.307-310 vol.2, 7-10 Oct 2001.

[4] Jing, Chen; Yongtian, Wang; Yue, Liu; Dongdong, Weng, "Navigating System for Endoscopic Sinus Surgery Based on Augmented Reality," *Complex Medical Engineering, 2007. CME 2007. IEEE/ICME International Conference on* , vol., no., pp.185-188, 23-27 May 2007

[5] F. Vahid and T.Givargis , "Introduction in *Embedded System Design: A Unified Hardware/Software Introduction*," John Wiley & Sons, Inc., New York, NY, 2001 pp 9-14

[6] G. J. Klinker, S. A. Shafer, and T. Kanade, "A phyisical approach to color image understanding," *International Journal of Computer Vision*, vol. 4, pp. 7-38, 1990.

[7] S.W. Lee and R. Bajcsy, "Detection of specularity using color and multiple views," *Proceedings of the 2nd European Conference on Computer Vision*, pp.99-114, Santa Margherita Ligure, Italy, May 1992.

[8] M. Celenk, "A Color Clustering Technique for Image Segmentation," *CVGIP*, pp.145-170, 1990.

[9] K. Schlüns and A. Koschan, "Global and Local Highlight Analysis in Color Images," *presented at Proc. 1st International Conference on Color in Graphics and Image Processing CGIP'2000, Saint-Etienne, France*, October 2000.

[10] C.A.Saint-Pierre, "Détection et correction des réflexions spéculaires dans les séquences d'images thoracoscopiques, " mémoire de maitrise, École polytechnique de Montréal, aout 2005

[11] M. Gröger, W. Sepp, T.Ortmaier, G. Hirzinger "Reconstruction of Image Structure in Presence of Specular Reflections" *Mustererkennung 2001: DAGM2001, Munich, Germany*, Sept 2001.

[12] F. Ortiz and F. Torres, "Automatic detection and elimination of specular reflectance in color images by means of MS diagram and vector connected filters," *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, vol.36, no.5, pp. 681-687, Sept. 2006.

[13] J.Angulo et J.Serra, "Colour image processing from luminance/saturation/hue in L1 representation" Revue *Traitement du Signal* , Vol. 21, No. 6, pp 583-604, December 2004.

[14] J. Bhattacharyya. "Detecting and removing specularities and shadows in images", mémoire de maitrise, Mc Gill University,Juin 2004

[15] A.C. Kokaram, R.D. Morris, W.J. Fitzgerald, P.J.W. Rayner, "Interpolation of missing data in image sequences," *IEEE Transactions on Image Processing 11(4), 1509-1519*, 1995.

[16] A. Hirani et T. Totsuka, "Combining Frequency and spatial domain information for fast interactive image noise removal," *Computer Graphics, pp. 269-276, SIGGRAPH 96*, 1996.

[17] S. Lin, "Generation of diffuse and specular appearance from photometric images," *Photometric Modeling for Computer Vision and Graphics, 1999. Workshop on.* , vol., no., pp.39-46, Aug 1999

[18] M. Bertalmio, G. Sapiro, C. Ballester and V. Caselles, "Image inpainting," *Computer Graphics, SIGGRAPH 2000,* July 2000.

[19] M. Bertalmio, A.L Bertozzi, G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting," *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on* , vol.1, no., pp. I-355-I-362 vol.1, 2001

[20] A. Downton and D. Crookes, "Parallel architectures for image processing ", *Electronics & Communication Engineering Journal* , vol.10, no.3, pp.139-151, Jun 1998.

[21] C. T. Johnston, K. T. Gribbon, D. G. Bailey, "Implementing Image Processing Algorithms on FPGAs," *Proc. of the 11th Electronics New Zealand Conference*, pp.118-123, November 2004.

[22] S. Sudha, G.R. Suresh, R. Sukanesh, "Wavelet Based Image Denoising Using Adaptive Thresholding," *Conference on Computational Intelligence and Multimedia Applications, 2007. International Conference on* , vol.3, no., pp.296-300, 13-15 Dec. 2007

[23] D.L. Donoho and I.M Johnstone, "Ideal spatial adaptation via wavelet shrinkage," *Biometrika,* vol. 81, pp. 425-455, 1994

[24] Pogossova, E.; Egiazarian, K.; Astola, J., "Signal denoising in tree-structured Haar basis," *Image and Signal Processing and Analysis, 2003. ISPA 2003. Proceedings of the 3rd International Symposium on* , vol.2, no., pp. 736-739 Vol.2, 18-20 Sept. 2003

[25] S. Tchoulack, J.M.P Langlois, F. Cheriet, "A video stream processor for real-time detection and correction of specular reflections in endoscopic images"*Proceedings of NEWCAS, June 2008*

Annexe

A Video Stream Processor for Real-time Detection and Correction of Specular Reflections in Endoscopic Images

Stéphane Tchoulack, J.M. Pierre Langlois and Farida Cheriet
Département de génie informatique et génie logiciel
École Polytechnique de Montréal, Canada
{ralph-stephane.tchoulack-ngounou, pierre.langlois, farida.cheriet}@polymtl.ca

Abstract— **This paper presents the architecture and FPGA implementation of a video processor for detection and correction of specular reflections in endoscopic images by using an inpainting algorithm. Stream processing and parallelism are used to exceed real-time performance on NTSC format video without the need for an external memory. The system was implemented in a XC2VP30 FPGA and uses 91% of available slices. Image quality is significantly enhanced.**

I. Introduction

Advances in video imaging have played a key role in bringing forth the widespread use of Minimally Invasive Surgery (MIS) in a variety of procedures such as cardiology, neurosurgery, orthopaedics, urology and oncology. However, the inherent difficulties of MIS techniques have traditionally imposed limitations on their applicability. Reduced instrumental control and freedom, combined with unusual hand-to-eye coordination and a limited view of the operating field, enforce restrictions on the surgeon and require considerable dexterity and skill. On the other hand, these procedures entail several benefits for the patient and the healthcare system: smaller incisions, minimal blood loss, preservation of normal tissue, reduced pain, and shortened recovery and rehabilitation times.

The use of video-assistance to facilitate MIS has been the focus of increasing attention since the early 1980s. In typical video-assisted MIS, a small camera called an endoscope is inserted into the surgical site via a small incision on the surface of the patient's body. The surgeon will exclusively use the endoscope video displayed on a monitor to view the surgical site and control the position of his instruments, which are also inserted through small incisions. Several sophisticated 3D navigation systems are under development in cardiology and neurosurgery. However, most of the current systems devised for spinal surgery still rely on rigidly fixed dynamic reference (or fiducial) markers on the instrumented vertebrae for the registration of preoperative patient data with the intra-operative data. This allows the surgeon to localize precisely the anatomical structures of interest while minimizing damage to adjacent critical structures.

Our team is working on the development of an augmented reality surgical environment using an image-based approach -- instead of using visible markers in the pre- and intra-operative images -- to achieve a non-contact, automated method for elastic 2D-3D registration. Unfortunately, the reflection of the light on specular surfaces such as metallic tools and moist tissues, as shown in Fig.1, produces artifacts in the images that render the task of automatic segmentation of endoscopic images very challenging. Improving the quality of endoscopic video is an important goal in itself, especially for augmented reality applications [8]. Any kind of processing must operate in real-time on regular sized video to be usable in a hospital, in an operating room. Previous work has shown that endoscopic images can be significantly improved, but with significant memory and processing requirements [1].

Figure 1. An endoscopic image with specular reflections [1]

In this paper, we present particularized algorithms for automated detection and correction of specular reflections in an endoscopic context, together with their real-time implementation in hardware. The paper is organized as follows. Section II presents the system architecture and some implementation considerations that affect algorithm development. Section III deals with the problem of detecting the specular reflections, and section IV is concerned with their correction. Section V presents results and a discussion.

II. System Architecture and Implementation Considerations

Fig. 2 shows the block diagram of our system. The input is a stream of pixels and synchronization signals from a video decoder connected to an endoscope. The video format is deinterlaced NTSC with an effective resolution of 720 × 480 pixels and a frame refresh rate of 60 Hz. The output includes pixel values and synchronization signals of the same format and refresh rate, transmitted to a VGA port for real-time display.

We require real-time video processing, which is difficult to achieve on a serial processor because of the great amount of data involved. In order to perform an operation on every pixel in real-time, the processor must execute 40 million operations per second. If we take into account the time to store and load data, the corresponding number of instructions rapidly increases. This is not outside the capabilities of FPGA implementations, however, where massive parallelism can be exploited. However, this choice implies special constraints related to the mode of processing data: streaming, offline and hybrid [2].

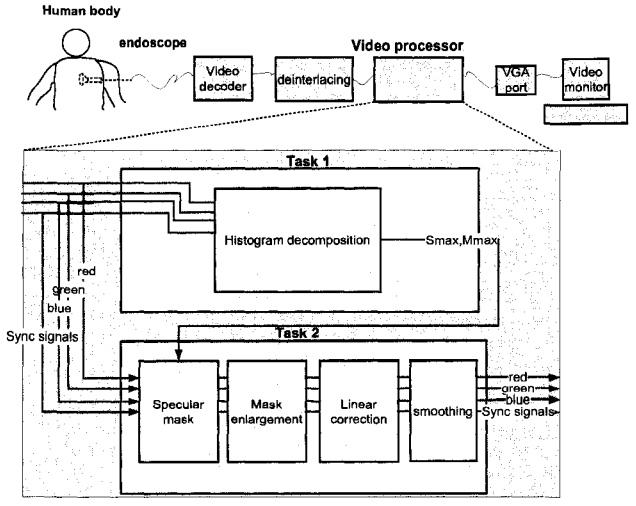


Figure 2. System block diagram

Storing one video frame requires approximately 1 MB. The limited amount of memory resources in a FPGA would normally require that an external memory be used, especially if several frames must be accessed for processing. However, this implies timing constraints related to the access time to the external memory. It must be shorter than the incoming rate of the pixels. Generally speaking, external memory doesn't allow multiple port access. The resultant memory bandwidth limitations make it desirable to avoid all external memories. Consequently, we favor the use of a streaming mode with as little memory storage as possible. We have to take the challenge to use a small amount of memory. In fact by using the internal memory of an FPGA we can avoid all the constraints due to the use of an external one.

There are two kinds of parallelism: data and pipeline [3]. Generally both of them are used at the same time: first we divide the frame into different sections (data parallelism), then each section passes through a processor which executes different tasks sequentially (pipeline parallelism). The algorithm described in this paper uses pipeline parallelism because each frame is entirely computed by one processor; but it's quite particular because two processors run on the same frame at the same time, and the results of one processor are used by the next one, as shown in fig. 2.

III. Detection of Specular Reflections

A diffuse reflection occurs when the incident ray is reflected in a multitude of angles. In this case, the incident energy is distributed in all the directions of the reflection. It generally occurs at the contact of a granular surface. That is the kind of reflection that permits us to see objects and their shape. A specular reflection, also called specularity, occurs when the incident light is reflected in only one direction. In this case, both the incident and reflected lights have the same energy, in principle without any loss. This energy can glow, especially when the light source is near the surface. This kind of reflection occurs when the surface is smooth. A reflection generally has both specular and diffuse components.

A. Histogram decomposition and criteria for detection

Specularities are by definition regions of an image where pixel intensity is very high and where the color matches the illumination source. Building an image histogram therefore has the potential to assist us in identifying these regions. Three separate histograms can be generated for each of the three colors in the image. For endoscopic images taken inside the body, the dominant color is red and the light source is white. Consequently, intensely white regions generally correspond to specular reflections. Analyzing the red, green and blue histograms for matching high intensity zones has the potential to point to specular regions [1].

It has also been shown that it is possible to use a grey-level image to detect specularities [4]. With a simple thresholding on this image we can detect specularities, because their pixel intensity is independent from other regions. Specularities are more visible in the S (saturation) component of the HSV plan. Consequently, two images are important to achieve good detection: the grey-level image and the saturation image.

The methods described in [1] and [4] use one-dimension histograms. They divide the image into two distinct regions: one where most of the pixels are located and the other where specularities are located. Generally however, these histograms tend to be noisy, which complicates the distinction between the two regions. In other terms, to perform a good thresholding on these histograms they must first be de-noised. Detecting a mass of pixels corresponding to specular reflections can be accomplished by double derivation of the histogram to extract the beginning and end of the specular region (in intensity) [1]. However, this involves many computations and it requires that a complete histogram of an image be stored.

It has also been suggested to use bi-dimensional histograms to perform detection. Specular reflection regions tend to be located in a static region of this histogram [5]. The bi-dimensional histogram is built as follows [6]:

$$m = \frac{1}{3}(r+g+b) \tag{1}$$

$$s = \begin{cases} \frac{1}{2}(2r-g-b) = \frac{3}{2}(r-m), \text{ if } (b+r) \geq 2g \\ \frac{1}{2}(r+g-2b) = \frac{3}{2}(m-b), \text{ if } (b+r) < 2g \end{cases} \tag{2}$$

where m is the intensity, s is the saturation, and r, g and b respectively represent the red, green and blue components of the image. Specularities can be identified from the bi-dimensional histogram based on the maximum values of m and s for the image [6]. They correspond to the region located in the lower right part of the M-S diagram. The relations proposed in

[6] tended to produce poor results in the context of endoscopic images. After careful investigation of the parameters with a large quantity of endoscopic images, we found that the following relations reliably identify pixels that are part of a specular reflection. A pixel p will be a part of the specular region if it meets the following conditions:

$$\begin{cases} m_p \geq \frac{1}{2} m_{\max} \\ s_p \leq \frac{1}{3} s_{\max} \end{cases} \quad (3)$$

where $m_{\max}$ and $s_{\max}$ are the maximum intensity of M and S for all pixels in an image, respectively.

Histogram decomposition corresponds to task 1 of our system (fig. 2). Once a pixel is received,

1. its M value is computed by (1);
2. having its M value, its S value is computed(2); and,
3. the $s_{\max}$ and $m_{\max}$ values are updated for each frame.

From a computation point of view, the approach using the bi-dimensional histogram is superior to the one with the one-dimension histogram. No de-noising or histogram post-processing is necessary. This eliminates a further cause of error in the form of rounding in fixed-point calculations. In fact, it is not required to store the bi- dimensional histogram at all, only to compute and track the maximum values of its two components for each frame. Simple thresholding is sufficient to detect specular regions.

B. *Composition of the specular mask*

The composition of the specular mask is part of task 2 in fig. 2; it works as follows. For each pixel p, the values of i_p and s_p are computed with (1) and (2). The relations of (3) are then evaluated to determine whether the pixel is part of the specular mask or not.

In theory, one would have to inspect all pixels from a frame f in order to calculate the $s_{\max}$ and $m_{\max}$ values, then apply (3) to that frame. However, we found that the $s_{\max}$ and $m_{\max}$ values vary little from frame to frame, and hence the values found for frame f allow to find the specular mask of the $f+1$ frame. The specularities tend to be more or less identical between two successive frames. This is true when there is no sudden change of direction of the camera or of the light source. In the worst case the error is limited to a single frame with duration under 17 ms. The obtained mask is a black and white frame, with the white parts indicating the presence of specularities.

C. *Mask enlargement*

The specular mask received from the detection includes only the specular spikes of the frame. It doesn't take into account the specular lobe or camera artifacts at the boundaries of specular regions and the diffuse regions, caused by the direction of the camera. If these components are not included in the specular mask, the correction will be severely compromised. Consequently, mask enlargement is necessary. One approach consists of using an intensity descent [1,5]. However, this algorithm requires a significant amount of memory access and is computationally intensive.

In order to accelerate computations, we propose the following process. The specular mask is inspected with the help of a sliding window [2]. We define a mask enlargement width of n pixels. When a specular pixel is encountered, all pixels within a $n \times n$ window centered on the specular pixel are included in the mask. The number N of buffered lines in the sliding window is given by:

$$N = 2n+1 \quad (4)$$

The width n of the enlargement depends on the width of the different artifacts we want to include into the mask. We have found that a value of $n = 3$ was adequate for most images. This means that there is a processing delay of 7 lines between the original mask and the enlarged one. Fig. 3 gives an example of a mask enlargement with $n = 1$. The white pixel represents the specular pixel, the bold lines represent the sliding window, and the light grey pixels represent the new specular pixels after successive iterations. The blue pixel is the actual pixel being computed.

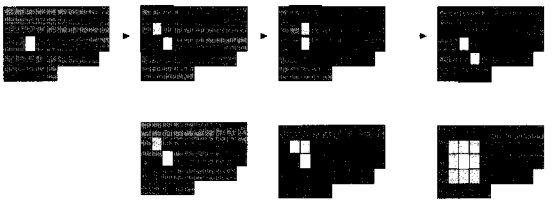


Figure 3. Mask enlargement with *n=1, up: input frame; down: output frame*

IV. CORRECTION OF SPECULAR REFLECTIONS

Correcting a frame consists of removing all the specularities previously detected and replacing them with information obtained from their neighborhood. One of the best ways to perform a good correction is to use an image in-painting algorithm, such as the Navier-Stokes algorithm [7]. However this kind of algorithm uses several loops to pass through a given frame. This requires large amounts of memory and computational effort. We aim to achieve a single frame memory architecture. We therefore propose the following approach, which operates line by line:

- An entire line of the frame is stored.
- For each specular region detected in this line, three data are stored: the value of the pixel before the specular region p_b the value of the pixel after the specular region p_e and the width of the specular region w.
- For each specular region, the linear skew a is calculated:

$$a = \frac{p_e - p_b}{w} \quad (5)$$

- The leftmost pixel in a specular region is given index zero. Pixel p_0 is given the value $p_b + a$. The corrected value of all other pixels is given by

$$p_{i+1} = p_i + a \quad (6)$$

At the end of the process there will be a delay of 1 line between the enlarged mask and the corrected frame.

Since linear correction operates only in the horizontal dimension, it is necessary to add correction along the vertical dimension. This is achieved by passing the corrected frame

through a smoothing window which replaces corrected pixels with the average of its neighbors. This is done by using a 3×3 sliding window with 5 passes. Each pass creates a delay of 3 lines, for a total of 15 lines delay.

The total delay of the system is 25 lines of 858 pixels each (with 720 active pixels); this delay of 0.8 ms between the non corrected input frame and the corrected output frame uses 64 KB of the internal memory. The delay is acceptable for real-time operation.

V. Results and Discussion

The system was first developed and implemented with Matlab to adjust parameters and processes and to build a baseline reference. This Matlab implementation included fixed point data types from the start. The system was then described at the register-transfer level with VHDL. Simulation and verification were performed with the help of Modelsim and an automated test bench.

Two systems were in fact implemented with different detection algorithms: the single [1] and bi-dimensional approaches. Table 1 presents resource usage for the implementation of each algorithm after the synthesis process. Bi-dimensional histogram detection uses 10 times fewer resources than the mono-dimensional version and it achieves better results.

TABLE I. Resources used for each hardware implementation on a Xilinx Virtex 2 pro XC2VP30 FPGA

algorithm	Flip-flops (27,392 available)	LUTs (27,392 available)	Brams (2,448 Kb available)
Mono-dimensional histogram detection	6,002 (21%)	10,531 (38%)	162Kb (6%)
Bi-dimensional histogram detection	571 (2%)	1,032 (3%)	108Kb (4%)
Correction	3,641 (13%)	24,044 (88%)	1,018Kb (42%)
Detection and correction	4,212 (15%)	25,076 (91%)	1,126Kb (46%)

Fig. 4 demonstrates the detection and correction of specular reflections. Hardware implementation of mono-histogram decomposition gives an unstable mask because of the computations needed to extract the beginning of the specular region.

The best results come from [7] because the correction is done on two dimensions, instead of one dimension. The specular regions are filled until there is no information to propagate from the boundaries. In other terms, the widths of the boundaries tend to zero. This needs a lot of memory to store multiple frames of the same picture.

The correction algorithm proposed in this paper works well when the specular region is entirely enclosed inside an object (Fig. 4). When it's located at the boundary of two different objects, the information coming from one object can be propagated to the other one.

After the optimization of placing and routing process, the system uses 91% of the available slices of a XC2VP30 FPGA. The maximum operating frequency is 32 MHz, which exceeds the minimum required of 27 MHz for real-time operation.

VI. Conclusion

This paper has presented a method and architecture to implement a processor able to detect and correct specularities in NTSC endoscopic videos. This is done with two parallel tasks in a streaming processing mode. Bi-dimensional histogram decomposition is computed to detect the specularities. It has the advantage of using few memory and computation resources without compromising the quality of the resulting image. The correction is done in two steps, a linear correction and a smoothing process. The system functions in real time and doesn't require an external memory.

References

[1] C. A. Saint Pierre, J. Boisvert, G. Grimard and F. Cheriet, "Dectection and Correction of Specular Reflections for Automatic Surgical Tool Segmentation in Thoracoscopic Images," *Machine Vision and Applications Journal* 2007, in press.

[2] C. T. Johnston, K. T. Gribbon and D. G. Bailey, "Implementing Image Processing Algorithms on FPGAs," *Proc. of the 11th Electronics New Zealand Conference*, pp.118-123, November 2004.

[3] A. Downton and D. Crookes, "Parallel architectures for image processing," *Electronics & Communication Engineering Journal*, vol.10, no.3, pp.139-151, Jun 1998.

[4] M. Gröger, W. Sepp, T. Ortmaier, G. Hirzinger "Reconstruction of Image Structure in Presence of Specular Reflections," Mustererkennung 2001: DAGM2001, Munich, Germany, Sept 2001.

[5] J. Bhattacharyya, "Detecting and Removing Specularities and Shadows in Images", Master's thesis, McGill University, June 2004.

[6] F. Ortiz and F. Torres, "Automatic detection and elimination of specular reflectance in color images by means of MS diagram and vector connected filters," *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, vol.36, no.5, pp. 681-687, Sept. 2006.

[7] M. Bertalmio, A.L. Bertozzi and G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting", Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol.1, pp I-355- I-362, 2001.

[8] Jing, Chen; Yongtian, Wang; Yue, Liu; Dongdong, Weng, "Navigating System for Endoscopic Sinus Surgery Based on Augmented Reality," *IEEE/ICME International Conference on Complex Medical Engineering*, pp.185-188, May 2007.

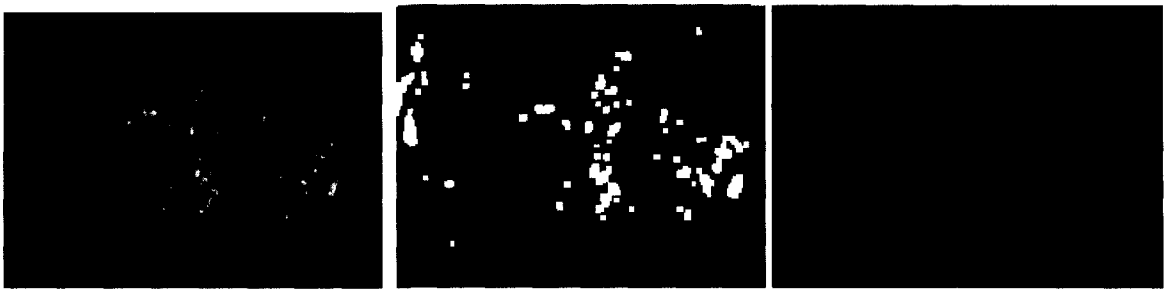


Figure. 4 Hardware implementation of the specular mask (middle) and the correction (right) of the left image